

COMP9727 Project Design Report

Name: Tianning Dong **zID:** z5559246

Project Title: SmartReads: Intelligent Book Recommendation System

1. Scope

1.1 Problem Statement

In today's information-rich environment, readers face the challenge of choosing the right book from an overwhelming number of options. Traditional platforms like Goodreads mainly rely on popularity or fixed user preferences, often resulting in generic and less personalized recommendations. Studies have shown that content-based methods offer higher personalization but lower accuracy, while popularity-based methods achieve better precision with less relevance to individual users (Kwan, Koh, & Jasser, 2020). These limitations hinder user engagement, especially in cold-start situations. To address this, we propose a personalized book recommendation system that combines collaborative filtering, content-based and knowledge-based approaches, enhanced by language models, to provide more relevant and explainable suggestions.

1.2 Target Users

The target readers of this system are those who seek personalized book suggestions without having to browse through an endless list of books. These users may:

- Want to explore new genres or authors
- Time is limited, so high-quality recommendations are needed
- Be new users with few interactions (cold start)
- Have a clear reading goal or topic in mind (such as science fiction)

1.3 Domain

This system belongs to the domain of digital book recommendation platforms, specifically implemented as a full-stack website with a Flask backend and a React frontend. It is designed to serve as a web-based book discovery platform where users can receive personalized book recommendations, interact with suggestion results, and manage their reading preferences through a modern and intuitive user interface.

1.4 Display & UI

Here is a mock up interface of the main page of my project:

Your Personalized Recommendations

Discover your next favorite book with our AI-powered recommendations tailored just for you.

The screenshot shows a web interface for 'SmartReads' with a navigation bar (Home, My Books, Preferences) and a search bar. The main section is titled 'Your Personalized Recommendations' and features five book cards. Each card displays the book title, author, a star rating, genre tags, a 'View on Goodreads' link, and interactive buttons for 'Like', 'Dislike', 'Rate', and 'To-Read'. The books recommended are 'The Seven Husbands of Evelyn Hugo' (4.5 stars), 'Atomic Habits' (4.7 stars), 'The Midnight Library' (4.2 stars), 'Educated' (4.6 stars), and 'The Silent Patient' (4.1 stars). A 'Load More Recommendations' button is located at the bottom of the card grid.

- Display Top-5 book recommendations per session
- Each card shows: Book title, author, rating, tags, and link to Goodreads page which users can see book detail

1.5 User Interaction

- Users can: Like, Dislike, Rate, Add to To-Read. I suppose define the four behaviour as follows:
 1. Like: A user clearly indicates their fondness for a certain book, providing a strong positive response as 5 star.
 2. Dislike: A user clearly indicates their dislike for a certain book, providing a strong negative response as 1 star.
 3. Rating: A detailed score given by users to a book (1 to 5 stars), which can reflect the intensity of interest.
 4. Want To Read: When a user expresses interest in a certain book but may not have read it, it is an implicit positive feedback.
- Users can check their preferences, select what books they want to read, and manually choose preferred tags or topics to further refine recommendation results.

These user actions serve as explicit and implicit feedback signals, which are logged and used to continuously improve the recommendation engine through retraining.

1.6 Updates & Cold Start

- **Cold-Start Users:** New users with no prior interactions are recommended books using **content-based filtering** and **knowledge-based filtering**. Users can optionally select preferred genres, book lengths, or authors. The system then matches these preferences against book metadata (e.g., `tags` , `authors` , `language_code`) to generate personalized suggestions.
- **Cold-Start Books:** New books with no user interactions are recommended based on their **content similarity** to popular or highly rated books. We compute similarity using features such as `tags` , `authors` , and `average_rating` . Books are also promoted if they match the preferences of users who recently rated similar items.
- **Model Updates:** The recommendation models (e.g., collaborative filtering and hybrid models) are **retrained weekly** using new ratings and "to-read" signals from user logs.
 - For scalable training, we use **batch retraining** for collaborative models and **incremental fitting** (e.g., `fit_partial` in LightFM) for hybrid models.
 - This periodic update ensures that the system adapts to evolving user preferences and integrates newly added content in a timely manner.

1.7 Business Model

I think my project can make profits in multiple ways:

- **Affiliate Marketing Integration** The system can generate outbound links (e.g., to Goodreads, Amazon, or online bookstores) using metadata like `goodreads_book_id` or `isbn` . When users purchase a book via these links, the platform can earn a commission through affiliate programs.
- **Premium Subscription Model** Users may subscribe to a premium version offering advanced features such as:
 - Unlimited personalized recommendations
 - Filtering by more advanced preferences (e.g., author nationality, mood)
 - Reading history analytics and insights
 - Early access to trending or upcoming book releases
- **Data Insights for Publishers** Anonymized interaction data (e.g., most viewed genres, books often added to "to-read" lists) can be used to generate reports for publishers and bookstores, providing them with market insights and helping them adjust marketing or publishing strategies.

2. Dataset

2.1 Dataset Source

- Dataset: Goodbooks-10k
- Source: <https://github.com/zygmuntz/goodbooks-10k>

2.2 Dataset Overview

- Users: 53,424
- Books: 10,000

The dataset I use includes the following core files:

- `ratings.csv` : Contains 5,976,479 user-book ratings in the form `user_id`, `book_id`, `rating` . Used for collaborative filtering. IDs are contiguous (users: 1–53424, books: 1–10000).
- `to_read.csv` : Contains 912,705 entries of users marking books as "to read" (implicit feedback). Format: `user_id`, `book_id` .
- `books.csv` : Metadata for each book, including:
 - **Identification:** `book_id` , `goodreads_book_id` , `best_book_id` , `work_id`
 - **Identifiers:** `isbn` , `isbn13`
 - **Content Metadata:** `title` , `original_title` , `authors` , `original_publication_year` , `language_code`
 - **Popularity Metrics:** `average_rating` , `ratings_count` , `work_ratings_count` , `ratings_1` , `ratings_2` , `ratings_3` , `ratings_4` , `ratings_5`
 - **Display Info:** `image_url` , `small_image_url`
- `book_tags.csv` : Maps each `goodreads_book_id` to a `tag_id` with a count indicating how many users assigned that tag to the book.
- `tags.csv` : Maps `tag_id` to human-readable `tag_name` .
- `books_xml.zip` : While `books_xml.zip` provides rich textual metadata (e.g., book description, user reviews), this project focuses only on structured data (`books.csv`, `ratings.csv`, `book_tags.csv`) for scalability and reproducibility. For the purpose of this project, I do not utilize the optional `books_xml.zip` metadata files, as the structured fields in `books.csv`, `book_tags.csv`, and `ratings.csv` are sufficient for building and evaluating our hybrid recommendation system.

2.3 Field Usage

- **Core Modeling Fields:**
 - `user_id` , `book_id` , `rating` : Used to construct the user-item interaction matrix for collaborative filtering.
- **Content-Based Features:**
 - `tags` , `authors` : Represent thematic and stylistic attributes of books to compute similarity.
 - `average_rating` , `ratings_count` : Indicate book popularity and quality, supporting ranking and cold-start scenarios.
 - `language_code` : Enable preference filtering.
- **Cold-Start Support:**
 - `tags` , `authors` : Enable recommendations for new users or books with no prior ratings.
 - `to_read.csv` : Provides implicit feedback to infer user interests even without explicit ratings.
- **Evaluation Support:**
 - `ratings.csv` : Serves as ground truth for offline evaluation.
 - Simulated interactions (like/dislike/to-read) complement traditional metrics with user-centric feedback.
- **Display and Linking:**
 - `image_url` , `small_image_url` : Used to display book covers in the UI.
 - `goodreads_book_id` : Enables hyperlinks to external Goodreads book pages for additional context.

2.4 Exploratory Analysis Summary

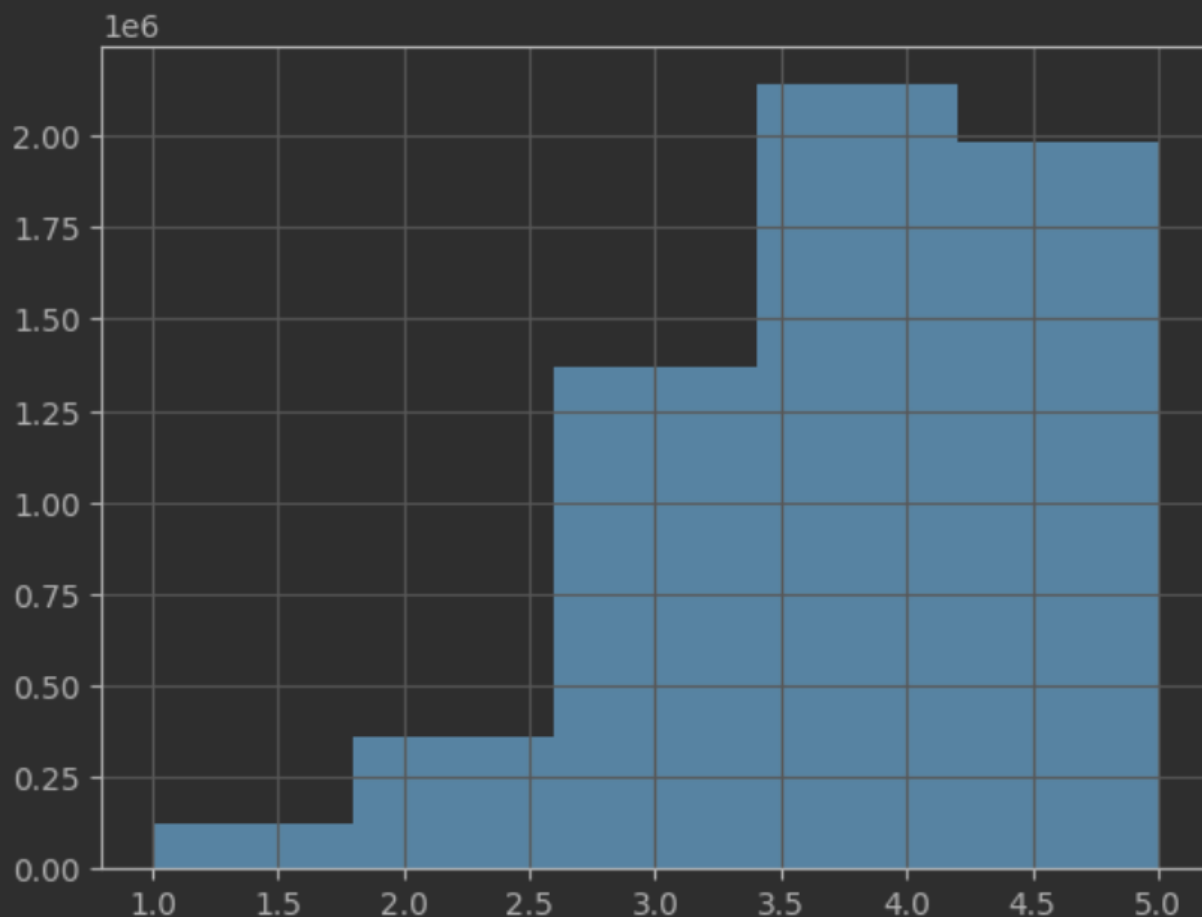
To better understand the dataset and inform system design, I conducted several exploratory analyses using Pandas:

1. Rating Distribution Skew

We examined the distribution of ratings to check user behavior bias:

```
r.rating.hist( bins = 5 )
```

<Axes: >



Observation: Ratings are heavily skewed toward 4 and 5 stars, indicating a strong positive bias, which may impact evaluation metrics like Precision or Recall.

2. Long-Tail Distribution of User and Book Activity

We calculated number of ratings per user and per book:

```
reviews_per_user = r.groupby('user_id').size()
reviews_per_user.describe()
```

```
reviews_per_book = r.groupby('book_id').size()
reviews_per_book.describe()
```

Observation: Users rated from 19 to 200 books (median=111). Books received 8–22806 ratings (median=248). Most books and users have relatively few interactions — classic long-tail distribution, requiring cold-start handling.

3. To-Read Implicit Feedback

I explored the `to_read.csv` file:

```
len(tr)
```

```
912705
```

Most books have been marked to read by somebody. Majority of the users have some books marked to read.

```
len(tr.book_id.unique())
```

```
9986
```

```
len(tr.user_id.unique())
```

```
48871
```

Observation: There are 912,705 to-read entries, covering 9,986 unique books and 48,871 users. This shows strong coverage across both users and items. The to-read signal is useful as implicit feedback to infer user intent in the absence of ratings.

4. Tag Diversity and Redundancy

We merged tag names with book tags:

```
bt = bt.merge(t, on='tag_id')
```

```
bt = bt.merge(b[['goodreads_book_id', 'title']], on='goodreads_book_id')
```

```
tag_counts = bt.groupby('tag_name').tag_name.count().sort_values(ascending=False)
```

```
tag_counts.head(20)
```

Observation: Top tags include "to-read", "favorites", "fiction", "owned-books" etc. High redundancy (e.g., "owned" vs "i-own") exists, indicating a need for cleaning or consolidation. Rich tag variety supports effective content modeling.

5. Completeness of Metadata

I verified key fields in `books.csv` :

```
b[['title', 'authors', 'average_rating', 'small_image_url']].isnull().sum()
```

	data
title	0
authors	0
average_rating	0
small_image_url	0

Observation: Most fields are complete, enabling reliable use in content-based recommendation.

These analyses confirm the dataset is rich enough for hybrid modeling, but also highlight the importance of addressing data sparsity, user bias, and tag redundancy.

2.5 Limitations

- Ratings and "to-read" actions lack timestamps, making it impossible to analyze temporal dynamics of user interests or implement time-aware recommendations.
- Tags are user-generated and contain redundancy, synonyms, misspellings, and low-frequency entries, requiring additional cleaning and normalization in my implementation.
- The rating distribution is heavily skewed towards high scores (4 and 5 stars), indicating positive user bias and potentially causing the system to overestimate the quality of popular books.
- The book_id refers to the "work" level rather than specific editions, which may mix ratings across different versions or translations, affecting recommendation accuracy.
- Despite the presence of "to-read" implicit feedback, some new users or niche books still suffer from sparse interactions, making cold-start a persistent challenge.

3. Method

3.1 Overview

I adopt a staged hybrid recommendation strategy:

1. Baseline: Collaborative filtering (User-based + Item-based)
2. Improved1: Content-based hybrid model
3. Improved2: Knowledge-based recommendation
4. Optional: LLM-enhanced explanation and semantic matching

3.2 Method 1: Baseline (User-based + Item-based Collaborative Filter)

- Tech Stack:
 1. Library: Surprise (Scikit-compatible library for recommender systems)
 2. Similarity Metric: Cosine similarity
 3. Algorithm: KNNBasic (UserKNN, ItemKNN)
 4. Input: ratings.csv (user_id, book_id, rating)
- Implement both user-based and item-based collaborative filtering:
 - **User-based CF**: recommends books liked by similar users, based on cosine similarity.
 - **Item-based CF**: recommends books similar to those the user rated highly, based on item-to-item similarity.
- Application in my Dataset:

Ratings matrix: user_id × book_id from ratings.csv

Predict how user u would rate item i by comparing with:

Other users who rated i (User-based CF)

Items rated by u (Item-based CF)

- Final prediction score: $\text{score}(u, i) = \alpha * \text{user_CF}(u, i) + (1 - \alpha) * \text{item_CF}(u, i)$

- α is tuned using cross-validation.
- Benefits: Provide a fast, interaction-driven baseline based solely on user ratings.
- Limitations: Cold-start problem (new users/items have no rating history). Ignores valuable metadata (tags, authors, language_code)

3.3 Method 2: Hybrid with Content

Hybrid Method = Collaborative Filtering + Content-Based Filtering

- Tech Stack:
 1. Library: LightFM
 2. Algorithm: Hybrid Matrix Factorization with BPR or Warp loss
 3. Input: User-item matrix: from ratings.csv or to_read.csv and other file includes Item features: tags, authors, average_rating, language_code
- Why This Method:
 1. Combines explicit interactions (ratings) with book metadata
 2. Learns latent embeddings from both behavior and content
 3. fit_partial() supports incremental model updates
 4. Handles implicit feedback from to_read.csv
 5. This choice is also supported by Mehendale (2024), who showed that LightFM's hybrid architecture effectively improves recommendation quality and mitigates cold-start issues in real-world systems through enhanced feature engineering and contextual embeddings.
- Application in my Dataset: Build item features matrix using: tags.csv + book_tags.csv → tag vectors , books.csv → one-hot encode authors, language_code

LightFM uses combined inputs to learn joint user-item embeddings

Cold-start books can be recommended using only their content features

- Improvements over Method 1:
 1. Can recommend books with no prior ratings if metadata is rich
 2. More robust to data sparsity
 3. Supports periodic online updates
- Limitations: Tag redundancy and sparsity may affect performance

3.4 Method 3: Knowledge-Based Recommendation

I use this try to address cold-start user scenario directly

- Implementation: Rule-based filtering using pandas and NumPy

UI Integration: Users optionally select preferences: genres (via tags), mood, language, author

Input: books.csv, tags.csv, user preference forms

- Application in my Dataset

1. Allow users to explicitly specify preferences such as:

- Genre (e.g., "historical fiction", "science fiction")
- Mood or reading purpose (e.g., "uplifting", "deep", "fun")
- Author style or nationality
- Language (language_code)

2. System filters books from books.csv using: language_code, authors, tags via joined book_tags.csv

3. Books matching all filters are ranked by average_rating

- Improvements over Method 2: Offers customization and transparency

Allows meaningful fallback when collaborative models fail

- Limitations: Relies on the quality and completeness of metadata

Does not leverage collaborative or latent relationships

3.5 Method 4: LLM-Enhanced Recommendation (Optional Extension)

LLM for Semantic Understanding and Explanation

- Tech Stack: Model: OpenAI GPT-3.5 via API Embedding: text-embedding-ada-002 for semantic similarity
- Why This Method:

GPT embeddings enable deep content understanding — detecting abstract themes like “dystopia” or “feminism” beyond surface-level tags

GPT excels at natural language generation, making recommendations more human-like and transparent

Useful for under-tagged or cold-start books, leveraging textual descriptions to find relevance

- Limitations: Requires access to GPT API

May need prompt design or caching for consistency

4. Evaluation (5 marks)

To comprehensively evaluate the effectiveness of the recommendation system, I will proceed from two aspects:

Model level: Measure the predictive ability of the recommendation algorithm on historical data;

System level: Through user behavior feedback and simulated questionnaire surveys, evaluate users' real experience and satisfaction.

4.1 Model level

I use the Top-5 recommendation list to predict for each user the five books they are most likely to like as the basic evaluation unit. All evaluation scores are calculated separately for each user and then averaged to reflect the personalized effect of the system.

Metric	Meaning
Precision@5	Proportion of the Top-5 recommended books that the user actually liked
Recall@5	Proportion of books the user liked that appear in the Top-5 recommendations
NDCG@5	Ranking-sensitive score: higher rewards if relevant books appear earlier
Coverage	Proportion of unique books recommended across users, reflecting diversity
Hit Rate@5	Whether at least one relevant book appears in the Top-5 recommendations

In this project, Precision@5 and NDCG@5 are chosen as the primary evaluation metrics. Precision@5 effectively measures the accuracy of recommendations — how many of the Top-5 results are actually liked by the user. NDCG@5 goes further by accounting for the ranking order, making it more aligned with real user experience. By contrast, Recall@5 is useful but more sensitive to how many books a user likes in total, especially with a short recommendation list like Top-5, so it is used as a secondary metric. Coverage and Hit Rate@5 are supplementary: they help assess the system's ability to explore diverse items and provide a broader view of recommendation success (Kaminskas & Bridge, 2017).

Data partitioning strategy:

I use the Leave-One-Out (LOO) strategy: For each user, retain one high-score rating (rating ≥ 4) as the test set, and the rest as the training set. If a user has only one rating record, ignore that user. Only users with a rating number of ≥ 10 will be evaluated to ensure the reliability of the training data. This strategy is close to the actual scenario, that is, predicting what the user's next favorite book will be.

4.2 System Metrics (User-Centric Evaluation)

To supplement the deficiency of offline indicators in user experience, I designed a small simulated user research experiment, which consists of two parts: behavioral records and subjective questionnaire feedback.

Behaviour evaluation metric table:

Metric	Meaning
Click-Through Rate (CTR)	Proportion of recommended items that the user clicked or liked
Skip Rate	Proportion of Top-5 recommendations that the user showed no interest in
Want-to-Read Rate	Percentage of recommended books that the user marked as “want to read” but hasn't read yet
New-to-Popular Ratio	Ratio of non-popular (less-rated) books to popular ones in recommendations, used to assess exploration capability

User case study survey:

Question	1 Strongly Disagree) 5 (Strongly Agree)
1. The recommended books match my interests.	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5
2. I discovered new books I would like to read	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5
3. The type of recommended book were diverse.	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5
4. The explanations increased my trust.	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5
5. I would continue using this book recommendation system.	<input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5

4.3 Real-Time Responsiveness & System Updates

Phase	Action	Tools
Offline Phase	Periodically train KNN and LightFM models; pre-generate GPT embeddings for books	Surprise, LightFM, OpenAI API
Online Phase	Update candidate pool in real-time based on user clicks; perform Top-N retrieval + ranking + explanation generation	Precomputed similarity matrix + GPT prompting

Update Strategy

- **Weekly batch retraining** for collaborative models (LightFM supports `fit_partial` for incremental updates).
- **GPT embeddings** are generated once per book and **cached in a vector database** to support efficient semantic search.

5. Milestone Timeline

Here is a plan I make for this project.

Week	Task	Notes
Week 6	Implement collaborative filtering (user-based & item-based CF)	Baseline recommender model
Week 7	Build hybrid recommendation model using LightFM	Combines ratings with content features
Week 8	Add knowledge-based filtering based on explicit user preferences	Enables customizable recommendations
Week 9	Design and implement evaluation metrics and process	<i>Optional: add LLM-enhanced features</i>
Week 10	Final integration, report writing, and presentation preparation	Finalization phase

6. References

1. Goodbooks-10k Dataset: <https://github.com/zygmuntz/goodbooks-10k>
2. LightFM Library: <https://making.lyst.com/lightfm/docs/>
3. Surprise Library for Collaborative Filtering: [[https://\(https://surpriselib.com/\)urpriselib.com/](https://surpriselib.com/)]
4. Kaminskas, M., & Bridge, D. (2017). Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems*, 7(1), 1–42. <https://doi.org/10.1145/2926720>
5. Mehendale, P. ENHANCING RECOMMENDATION SYSTEMS WITH ADVANCED FEATURE ENGINEERING.
6. Kwan, C., Koh, M. Q., & Jasser, M. B. (2020). A comparison study between content-based and popularity-based filtering via implementing a book recommendation system. *International Journal Of Advanced Research In Engineering & Technology*, 11, 12.