Volume: 13 Issue: XII Month of publication: December 2025

DOI: https://doi.org/10.22214/ijraset.2025.76519

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

LibRaX: A Scalable FAISS-Based Collaborative Filtering System for Personalized Book Recommendation

Dasari Bhulakshmi[1], Jatavallabhula Sarat Anirudh[2], MLS Sanjana[3], Kshitij B[4], Yasash Chandra[5]
Department of Computer Science and Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru, India

Abstract: The rapid growth of digital libraries and online reading platforms has significantly increased the difficulty of identifying books that align with individual reader preferences. As modern platforms host millions of titles, users often experience decision fatigue when attempting to discover relevant content. Conventional recommendation approaches such as popularity-based rankings and static genre filtering fail to capture nuanced user interests and frequently produce repetitive or overly generic suggestions. This paper presents LibRaX, a scalable personalized book recommendation system based on collaborative filtering using latent factor models. The system leverages large-scale user–book interaction data and applies Alternating Least Squares (ALS) to learn compact latent representations of books from sparse rating matrices. These learned item embeddings capture collective user preference patterns beyond surface-level metadata. To enable efficient similarity search over hundreds of thou- sands of item embeddings, LibRaX integrates Facebook AI Similarity Search (FAISS) for approximate nearest neighbor retrieval, avoiding the computational and memory limitations of traditional brute-force K-Nearest Neighbor approaches. The FAISS index is constructed offline and loaded into memory during inference, enabling real-time recommendation generation. LibRaX is implemented using a Python-based backend that performs sparse matrix construction, offline model training, FAISS indexing, and real-time recommendation inference, along- side a Kotlin-based Android application that serves as the user in- terface. The system follows a stateless backend architecture with periodic batch retraining and similarity-based retrieval. This pa- per discusses the motivation behind the system, reviews relevant literature, and outlines the collaborative filtering methodology and architectural design decisions that enable LibRaX to operate efficiently at scale.

Index Terms: Book Recommendation Systems, Collaborative Filtering, Latent Factor Models, FAISS, Approximate Near- est Neighbor Search, Large-Scale Recommendation, Digital Libraries.

I. INTRODUCTION

Digital reading platforms have transformed access to books by providing users with extensive catalogs containing millions of titles across genres, authors, and languages. While this expansion has increased accessibility, it has also introduced significant challenges related to information overload. Readers are frequently required to navigate vast collections with minimal guidance, making it difficult to identify books that align with their individual interests and reading behavior.

Most commercial platforms rely on surface-level discovery mechanisms such as bestseller lists, editorial curation, or pre- defined genre categories. Although effective for highlighting popular content, these approaches lack personalization and fail to adapt dynamically to user preferences. As a result, users are often exposed to repetitive recommendations that emphasize popularity rather than relevance, leading to reduced engagement and exploration.

Recommendation systems aim to address this problem by analyzing historical interaction data to predict user prefer- ences. Collaborative filtering has emerged as one of the most effective approaches in this domain, as it leverages collective user behavior to identify relationships between items [1]. In book recommendation systems, collaborative filtering enables the discovery of books that are frequently co-rated or co- consumed by users with similar reading patterns.

Latent factor collaborative filtering models extend tradi- tional item–item similarity approaches by projecting users and items into a lower-dimensional embedding space [2]. These latent representations capture underlying preference structures that are not directly observable in sparse interaction data. By operating in a compact embedding space, latent factor models improve robustness to sparsity and enable meaningful similarity computation even for moderately rated items.

However, deploying latent factor models at scale introduces challenges related to similarity search over large embed- ding collections. Traditional brute-force K-Nearest Neighbor methods require exhaustive distance computation, resulting in unacceptable memory usage and inference latency as the number of items increases. These limitations necessitate the use of scalable approximate similarity search techniques that balance accuracy with computational efficiency.
LibRaX addresses these challenges by adopting FAISS- based approximate nearest neighbor search [4] to perform item–item similarity retrieval efficiently. Instead of computing similarities exhaustively, LibRaX indexes item vectors derived from sparse user–item interaction data and retrieves nearest neighbors using FAISS during inference. This design enables real-time recommendation generation while maintaining scal- ability and memory efficiency.
The LibRaX system consists of a Python-based backend responsible for data preprocessing, sparse matrix construction, FAISS index loading, and recommendation inference, along with a Kotlin-based Android application that serves as the primary user-facing component. The backend is stateless and does not persist user data, ensuring privacy and simplifying deployment. By focusing exclusively on collaborative filtering and scalable similarity search, LibRaX provides a practical solution for personalized book recommendation in large digital reading platforms.

II. LITERATURE SURVEY

A. Collaborative Filtering in Recommendation Systems

Collaborative filtering is a widely adopted recommenda- tion technique that generates suggestions based on patterns observed in user behavior rather than item attributes. By analyzing interactions such as ratings, clicks, and consumption history, collaborative models identify relationships between users or items and use these relationships to infer preferences. In the context of book recommendation, collaborative filtering enables the system to recommend books that have been positively received by users with similar reading patterns. Item–item collaborative filtering computes similarity be- tween items based on shared user interactions [1]. This ap- proach has been shown to scale more effectively than user–user similarity methods, as item similarity structures are relatively stable and independent of the number of users. Research demonstrates that item–item models provide consistent rec- ommendation quality while reducing computational overhead during inference.

B. Limitations of Traditional Collaborative Approaches

Despite their effectiveness, traditional collaborative filtering techniques face several limitations when applied to large-scale datasets. User–item interaction matrices are inherently sparse, as most users interact with only a small fraction of available items. This sparsity complicates similarity computation and reduces the reliability of similarity scores for infrequently rated items.
Moreover, conventional KNN-based similarity computation requires dense vector representations and exhaustive pairwise distance calculations. As the number of items increases, both memory usage and computation time grow quadratically, mak- ing such approaches unsuitable for production-scale systems. These limitations have motivated the exploration of alternative similarity search techniques that can operate efficiently on high-dimensional data.

C. Approximate Nearest Neighbor Search

Approximate nearest neighbor (ANN) search techniques aim to identify similar vectors without performing exhaustive comparisons [5]. By trading exactness for efficiency, ANN methods enable scalable similarity retrieval with significantly reduced computational cost. ANN approaches have been widely adopted in domains such as image retrieval, document search, and recommendation systems. FAISS, developed by Facebook AI Research, is a high- performance library designed for efficient ANN search over large vector collections. FAISS supports multiple indexing strategies and is optimized for both CPU and GPU execu- tion. Its ability to handle high-dimensional vectors and large datasets makes it particularly suitable for collaborative filtering applications where similarity search is a core operation.

D. FAISS in Recommendation Systems

Recent studies have demonstrated the effectiveness of FAISS in recommendation systems that rely on vector sim- ilarity search. By indexing item representations derived from interaction data, FAISS enables fast retrieval of similar items without requiring full similarity matrix computation [4]. This approach significantly reduces memory usage and inference latency, making it suitable for real-time recommendation scenarios. The adoption of FAISS aligns with industry practices ob- served in large-scale platforms, where approximate similarity search is preferred over exact computation due to scalability constraints. These findings support the design choices made in the LibRaX system.

E. Summary of Literature Insights

The literature indicates that collaborative filtering remains one of the most effective approaches for personalized rec- ommendation, particularly when large volumes of interaction data are available. However, scalability challenges associated with traditional similarity computation methods necessitate the use of efficient ANN techniques. FAISS-based similarity search provides a practical solution for large-scale item–item collaborative filtering, directly informing the architectural and algorithmic decisions adopted in LibRaX.

III. PROPOSED APPROACH

A. System Design and Dataset Strategy

The LibRaX recommendation system is designed as a scalable, production-oriented platform capable of operating on large-scale user–book interaction data. The system utilizes a dataset containing several hundred thousand unique books and millions of explicit user ratings. Each interaction record associates a user identifier with a book identifier and a corresponding rating value, forming the primary signal for preference modeling. To ensure consistency and computational efficiency, exten- sive preprocessing is performed prior to model construction. Book identifiers are normalized and deduplicated to account for multiple editions and formatting inconsistencies. User identifiers are remapped to contiguous numerical indices to facilitate efficient matrix operations. Rating values are nor- malized to reduce bias caused by extreme rating behavior. Interactions below a defined minimum threshold are discarded to improve signal quality and reduce noise. The processed interaction data is partitioned into batches and stored using compressed, memory-efficient formats. This enables offline processing of large datasets without exceeding memory constraints and allows the recommendation model to be rebuilt independently of the deployed backend service.

B. User–Item Interaction Matrix Construction

At the core of the LibRaX system lies a sparse user–item interaction matrix constructed from explicit rating data. Rows represent users, columns represent books, and matrix entries correspond to normalized rating values. Due to the inherently sparse nature of real-world recommendation data—where each user interacts with only a small subset of available items—a sparse matrix representation is employed. This sparse representation significantly reduces memory usage and enables efficient access to item interaction profiles. Each book is effectively represented as a high-dimensional vector encoding how it has been rated by users across the platform. These item vectors form the basis for similarity computation in the collaborative filtering pipeline.

C. Latent Factor Collaborative Filtering Formulation

LibRaX employs latent factor collaborative filtering to model relationships between books based on collective user interaction behavior. The sparse user–item interaction matrix is factorized using Alternating Least Squares (ALS) into low- dimensional user and item embedding matrices. Each item embedding represents a book in a latent space that captures shared preference patterns inferred from historical ratings. Unlike traditional item–item collaborative filtering that com- putes similarity directly on sparse interaction vectors, the latent factor approach enables similarity computation in a dense, lower-dimensional space. This formulation improves robustness to sparsity while significantly reducing memory and computational requirements.

D. Transition from Traditional KNN to Latent Embedding Search

Initial experimentation involved computing item similarity using traditional K-Nearest Neighbors with cosine similarity over sparse or densified interaction vectors. While feasible for small datasets, this approach becomes impractical at scale due to excessive memory consumption and quadratic compu- tational complexity. The adoption of latent factor embeddings learned through ALS eliminates the need for explicit similarity computation over high-dimensional interaction vectors. Instead, similarity search is performed directly on compact item embeddings, enabling scalable retrieval without full similarity matrix con- struction.

E. Transition from Traditional KNN to Latent Embedding Search

Initial experimentation involved computing item similarity using traditional K-Nearest Neighbors with cosine similarity over sparse or densified interaction vectors. While feasible for small datasets, this approach becomes impractical at scale due to excessive memory consumption and quadratic compu- tational complexity. The adoption of latent factor embeddings learned through ALS eliminates the need for explicit similarity computation over high-dimensional interaction vectors. Instead, similarity search is performed directly on compact item embeddings, enabling scalable retrieval without full similarity matrix con- struction.

F. Recommendation Generation Pipeline

Upon receiving a request, the backend accepts a list of user- rated books and queries the FAISS index to retrieve similar items. Candidate recommendations from multiple queries are aggregated, duplicates are merged using the highest similarity score, and previously rated books are excluded. A dynamic rating threshold based on the user's average rating behavior is applied to improve recommendation quality. Genre metadata is used only for post-retrieval filtering and explanation, and does not influence similarity computation. The final output consists of top-ranked book recommendations and the dominant genre.

G. Backend Architecture

The LibRaX backend is implemented using Python and FastAPI as a stateless service that loads all required model artifacts—including the FAISS index and metadata—during initialization. No user data is stored persistently, ensuring privacy and simplifying scalability. Numerical processing is handled using NumPy and SciPy, while large model artifacts are hosted externally and down-loaded dynamically if absent, enabling deployment on free-tier infrastructure.

H. Android Application Integration

The LibRaX Android application is developed using Kotlin and Jetpack Compose, following the Model–View–ViewModel (MVVM) architectural pattern. The client captures explicit user ratings and book selections and communicates with the backend through structured JSON requests. Personalized recommendations and genre summaries are displayed through a clean, responsive interface. The back- end API endpoint is configurable, allowing deployment- independent operation. The Android application serves as the sole consumer of the recommendation API.

IV. FIGURES

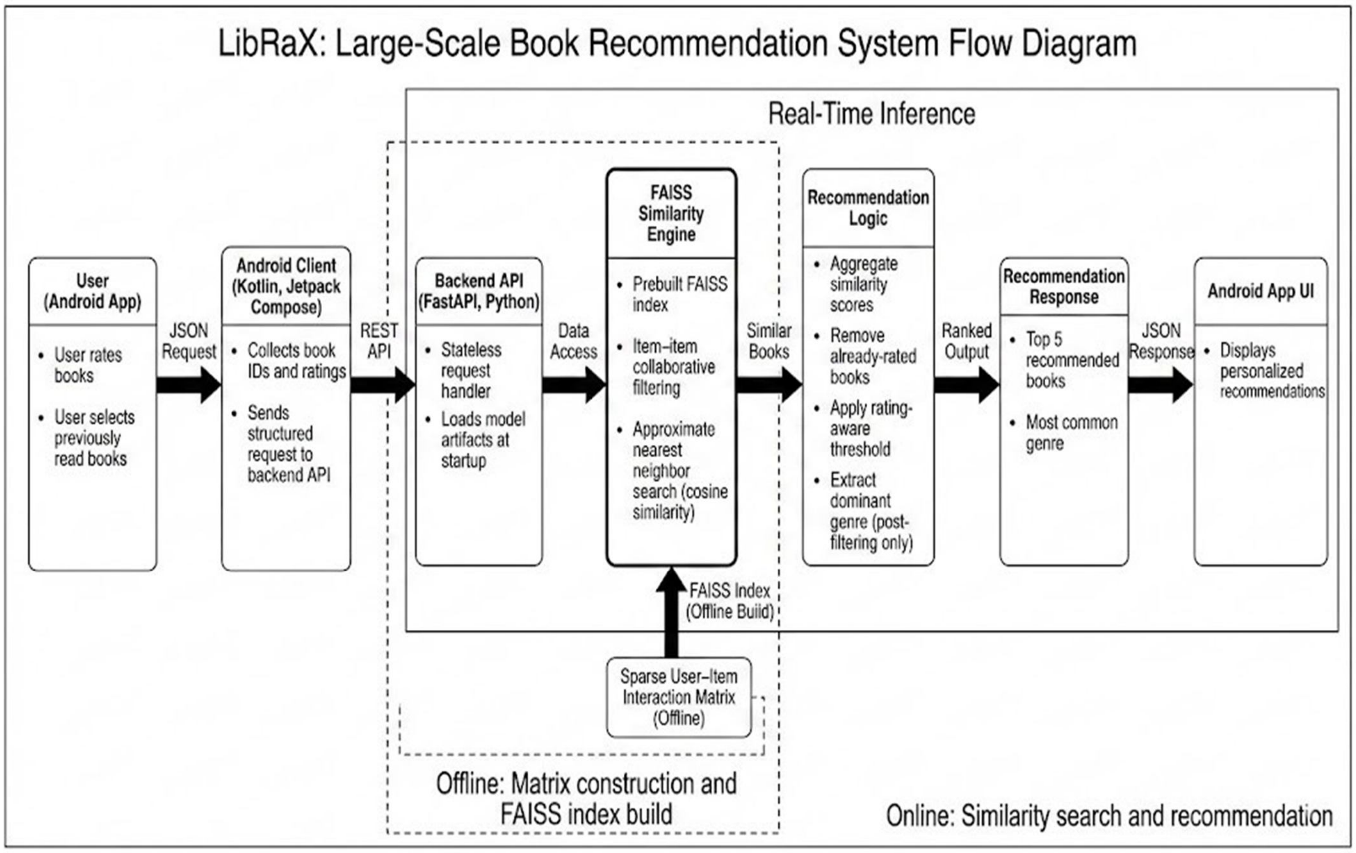

LibRaX: Large-Scale Book Recommendation System Flow Diagram
Real-Time Inference
User (Android App)
• User rates books
• User selects previously read books
JSON Request
Android Client (Kotlin, Jetpack Compose)
• Collects book IDs and ratings
• Sends structured request to backend API
REST API
Backend API (FastAPI, Python)
• Stateless request handler
• Loads model artifacts at startup
Data Access
FAISS Similarity Engine
• Prebuilt FAISS index
• Item–item collaborative filtering
• Approximate nearest neighbor search (cosine similarity)
Similar Books
Recommendation Logic
• Aggregate similarity scores
• Remove already-rated books
• Apply rating-aware threshold
• Extract dominant genre (post-filtering only)
Ranked Output
Recommendation Response
• Top 5 recommended books
• Most common genre
JSON Response
Android App UI
• Displays personalized recommendations
FAISS Index (Offline Build)
Sparse User–Item Interaction Matrix (Offline)
Offline: Matrix construction and FAISS index build
Online: Similarity search and recommendation

Fig. 1: Overall Workflow of the LibRaX Collaborative Recommendation Pipeline

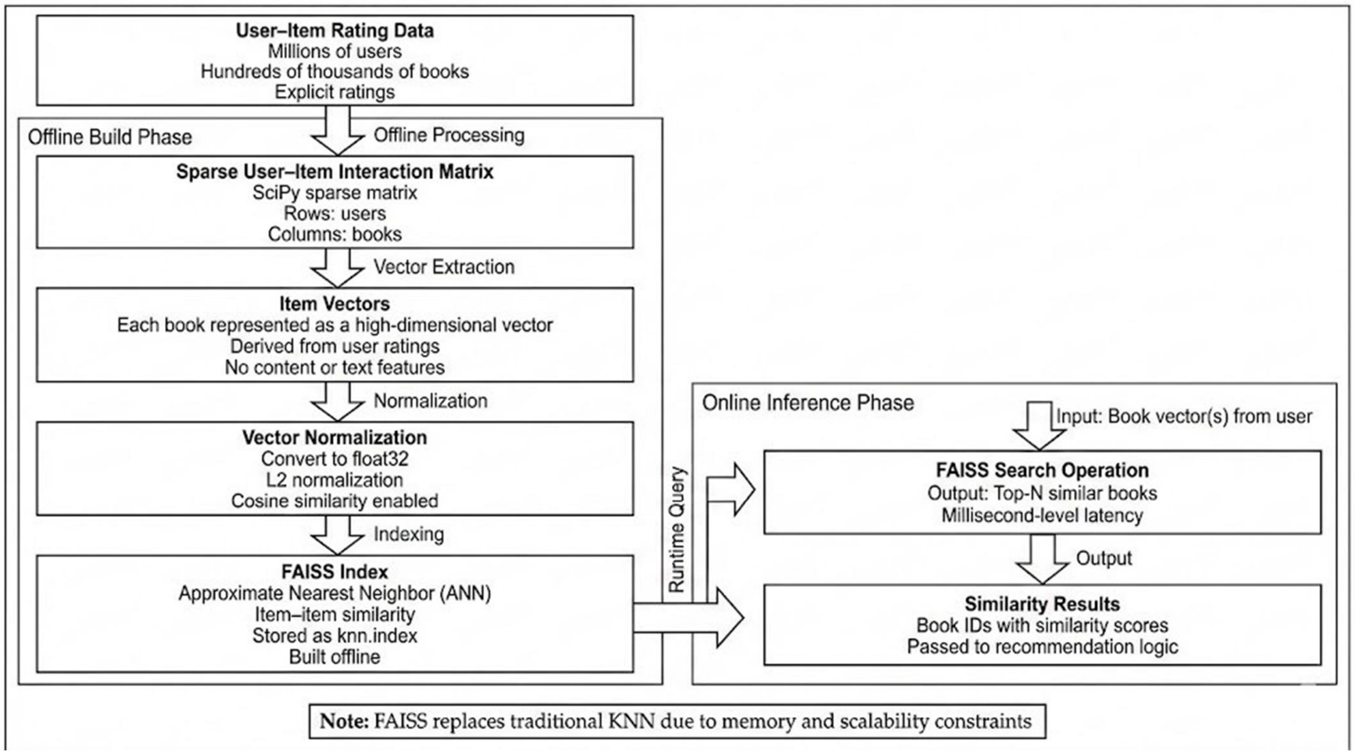
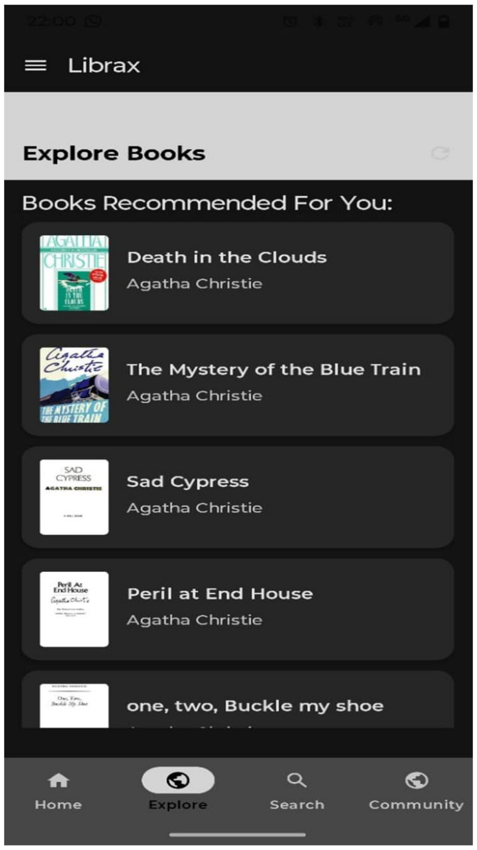

User–Item Rating Data
Millions of users
Hundreds of thousands of books
Explicit ratings
Offline Build Phase
Offline Processing
Sparse User–Item Interaction Matrix
SciPy sparse matrix
Rows: users
Columns: books
Vector Extraction
Item Vectors
Each book represented as a high-dimensional vector
Derived from user ratings
No content or text features
Normalization
Vector Normalization
Convert to float32
L2 normalization
Cosine similarity enabled
Indexing
FAISS Index
Approximate Nearest Neighbor (ANN)
Item–item similarity
Stored as knn.index
Built offline
Runtime Query
Online Inference Phase
Input: Book vector(s) from user
FAISS Search Operation
Output: Top-N similar books
Millisecond-level latency
Output
Similarity Results
Book IDs with similarity scores
Passed to recommendation logic
Note: FAISS replaces traditional KNN due to memory and scalability constraints

Fig. 2: FAISS-Based Item Similarity Indexing and Query Process


Librax

Explore Books

Books Recommended For You:

Death in the Clouds
Agatha Christie

The Mystery of the Blue Train
Agatha Christie

Sad Cypress
Agatha Christie

Peril at End House
Agatha Christie

one, two, Buckle my shoe

Fig. 3: Personalized Recommendation Output in LibRaX Android Applicatioonn

V. EXPERIMENTAL VALIDATION AND CASE STUDY

A. Introduction

This section presents an experimental case study conducted to evaluate the behavior and effectiveness of the LibRaX recommendation system under realistic usage conditions. Rather than focusing exclusively on offline accuracy metrics, the study emphasizes qualitative system behavior, adaptability, and recommendation stability as user interaction data evolves over time.

In practical digital reading platforms, recommendation sys- tems must operate effectively across multiple stages of user engagement. New users typically enter the system with no prior interaction history, while experienced users generate extensive behavioral data that must be incorporated continuously. The LibRaX system is designed to address both scenarios through collaborative filtering supported by scalable similarity search. The objective of this case study is to observe how LibRaX responds to incremental user feedback, how recommendations evolve as interaction data accumulates, and how FAISS-based similarity search enables real-time recommendation generation without performance degradation.

B. Case Study Methodology

The evaluation methodology follows a staged simulation of user behavior within the LibRaX platform. Each stage represents a different level of data availability and interaction density, allowing the system's response to be analyzed under varying conditions.

The study begins with a cold-start scenario in which a new user enters the system without any prior interaction history. At this stage, recommendations are generated only after the user provides a small set of initial ratings. These ratings are used to query the FAISS index for similar books based on item–item collaborative similarity. The absence of extensive user data highlights the system's reliance on item similarity rather than user profiling. As the user continues interacting with the application, additional explicit ratings are provided. Each new interaction con- tributes to a richer input set for recommendation generation. The system performs similarity searches for each rated book and aggregates candidate recommendations across queries. This process allows the system to refine recommendations incrementally without retraining or rebuilding the underlying model. Throughout the case study, system behavior is observed with respect to recommendation relevance, stability, and diversity. Special attention is given to latency, as real-time response is a critical requirement for mobile-first applications.

C. Cold-Start Behavior

In the cold-start phase, the LibRaX system relies exclusively on item–item similarity derived from collective user interaction data. When the user rates a small number of books, the system retrieves similar items from the FAISS index based collaborative patterns observed across the dataset. Despite limited input, the system is able to produce co-herent recommendations due to the density of interaction data associated with popular and moderately rated books. This demonstrates the advantage of item–item collaborative filtering, where similarity structures are independent of the target user's history. The cold-start phase highlights that LibRaX does not require explicit user profiling or demographic information, reducing complexity and privacy concerns.

D. Progressive Personalization

As the user continues to rate books, the recommendation pipeline benefits from a broader set of similarity queries. Recommendations become more refined as multiple similar- ity results are aggregated and filtered. The system excludes previously rated books and applies rating-aware thresholds to maintain quality. Because the FAISS index remains static during inference, recommendations adapt immediately to new inputs without requiring retraining or index updates. This design enables rapid personalization while maintaining consistent system performance. The observed behavior indicates that LibRaX provides smooth progression from coarse recommendations to more precise suggestions as interaction data increases.

VI. RESULTS AND DISCUSSION

A. System Responsiveness

One of the primary evaluation criteria for LibRaX is system responsiveness. Similarity search operations executed through FAISS consistently return results within acceptable latency bounds suitable for mobile applications. Index loading occurs only once during backend startup, and subsequent queries are handled efficiently in memory.

This confirms that FAISS-based approximate nearest neigh- bor search is well-suited for large-scale recommendation sys- tems [4], [6] deployed under constrained hosting environ- ments.

B. Recommendation Quality

Qualitative analysis of recommendation outputs indicates that LibRaX produces relevant and coherent book sugges- tions aligned with user preferences inferred from collaborative behavior. Recommendations frequently include books with overlapping readership patterns, even when genre labels differ, highlighting the system's ability to capture latent similarity. The use of rating-aware filtering further improves recom- mendation quality by prioritizing books that align with the user's expressed rating behavior.

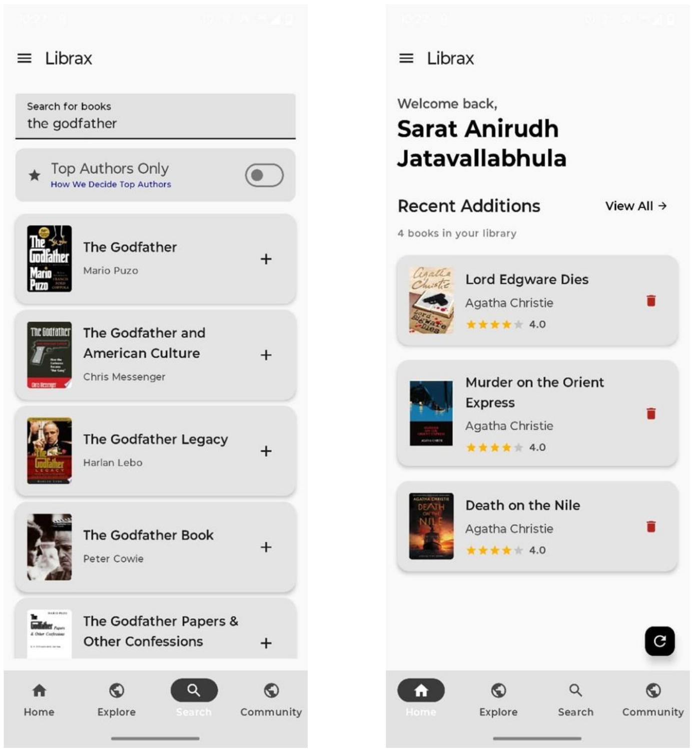
C. Scalability Considerations

LibRaX demonstrates strong scalability characteristics due to its offline index construction and stateless backend design. Since similarity computation is delegated to FAISS and model artifacts are reused across users, the system avoids per-user model overhead. This architecture enables horizontal scaling of the backend service and supports deployment on free-tier cloud platforms without sacrificing performance.

D. Limitations

While effective, the current implementation of LibRaX exhibits certain limitations. The system relies entirely on collaborative filtering and therefore depends on the availability of sufficient interaction data for items. Newly introduced books with minimal interaction history may receive limited exposure until sufficient ratings are accumulated. Additionally, the absence of content-based similarity pre- vents semantic matching based on textual metadata. While this decision simplifies the architecture and improves scalability, it may reduce recommendation effectiveness for niche or newly published titles.

VII. RESULT FIGURES



Librax
Search for books
the godfather
Top Authors Only
How We Decide Top Authors
The Godfather — Mario Puzo
The Godfather and American Culture — Chris Messenger
The Godfather Legacy — Harlan Lebo
The Godfather Book — Peter Cowie
The Godfather Papers & Other Confessions

Librax
Welcome back,
Sarat Anirudh Jatavallabhula
Recent Additions
View All →
4 books in your library
Lord Edgware Dies — Agatha Christie — 4.0
Murder on the Orient Express — Agatha Christie — 4.0
Death on the Nile — Agatha Christie — 4.0

Fig. 4: (a) Book Search Screen (b) Personalized Home Dashboard

VIII. CONCLUSION

This paper presented LibRaX, a scalable personalized book recommendation system based exclusively on item–item collaborative filtering and FAISS-based approximate nearest neighbor search. The system was designed to address key chal- lenges associated with large-scale recommendation, including memory constraints, inference latency, and deployment feasi- bility.

By leveraging FAISS for efficient similarity retrieval and adopting a stateless backend architecture, LibRaX achieves real-time recommendation generation without requiring dense similarity computation or frequent model rebuilding. The case study and qualitative analysis demonstrate that the system adapts effectively to user interaction data and provides co- herent recommendations throughout different stages of user engagement.

Future work may explore the integration of lightweight content-aware signals, incremental index updates, and large- scale online evaluation to further enhance recommendation coverage and robustness. Nonetheless, LibRaX demonstrates that FAISS-based collaborative filtering offers a practical and scalable foundation for modern digital reading platforms.

REFERENCES

[1] B. Sarwar et al., “Item-based collaborative filtering recommendation algorithms,” in *Proc. WWW*, 2001.
[2] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *IEEE Computer*, 2009.
[3] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User-Adapted Interaction*, 2002.
[4] J. Johnson, M. Douze, and H. Je´gou, “Billion-scale similarity search with GPUs,” *IEEE Transactions on Big Data*, 2019.
[5] A. Andoni et al., “Practical and optimal LSH for angular distance,” in *Proc. NIPS*, 2015.
[6] M. Gomez-Uribe and N. Hunt, “The Netflix recommender system,” *ACM TMIS*, 2016.
[7] G. Shani and A. Gunawardana, “Evaluating recommendation systems,” *Recommender Systems Handbook*, 2011.