

EXPLORING THE POTENTIAL OF A GENERATIVE AI FOR BOOKS RECOMMENDATION SYSTEM FOR UNIVERSITY STUDENTS

Abboskhon Bokikhonov

*Inha University in Tashkent, School of Computer and Information Engineering
Undergraduate Student, B.Sc. (in progress), Information and Communication Engineering
Mirshod Makhsudov*

*Inha University in Tashkent, School of Computer and Information Engineering
Undergraduate Student, B.Sc. (in progress), Information and Communication Engineering
Samobek Erkinov*

*Inha University in Tashkent, School of Computer and Information Engineering
Undergraduate Student, B.Sc. (in progress), Information and Communication Engineering*

Abstract. This research paper presents the design of a Books Recommendation Database Management System (Books Recommendation DBMS) for university libraries and student portals. Most existing recommenders suggest only one book, which is not enough in an academic setting where one topic may be covered better across chapters of several textbooks. To close this gap, we combine a relational database, an Extended Entity-Relationship model, an SQL implementation, and a Generative AI layer that returns multi-book recommendations tuned to the reader's major, year of study, GPA and topic interests. The paper covers data specification and logical requirements, conceptual and logical design, the physical schema built in MySQL Workbench, functional requirements, system architecture, and a small set of SQL experiments that validate the database. Findings on collaborative filtering, k-Nearest Neighbors, Singular Value Decomposition and Stochastic Gradient Descent shaped the recommendation logic and the cold-start strategy. The result is a modular base that can be extended later with more AI and NLP components.

Keywords: book recommendation system; database management system; generative AI; collaborative filtering; entity-relationship model; SQL; MySQL; matrix factorization; university library; personalized recommendation.

1. Introduction and Scope

In this research paper we study book recommendation systems and describe our product, Books Recommendation DBMS, which can be embedded into other software to improve its functionality. On large online bookshops, users still spend a lot of time looking for a suitable book. The reasons vary: books are organized in the database by business priorities rather than user preferences, the needed book is missing from the inventory, and so on. University libraries face a similar problem. They often give students only one or two textbooks per subject, even though some subtopics are explained better in chapters of other books. For instance, in a Database course Silberschatz's textbook is used for the whole course, while the design of database management systems is explained particularly well in Elmasri's textbook. The two books also use different notations, so reading both gives a wider view. Most existing recommenders, however, return only a single book.

Contributions of this work:

Research about books in general — who the readers are, who the target audience of our application is, and what the limits and real-world uses of the Books Recommendation DBMS are. This part looks at both business rules and scientific evidence about reading preferences and library management.

Knowledge-based analysis of data requirements in (subject, predicate, object) form as the first step of our Software Requirements Specification (partially following IEEE Std 830-1998).

We relied on the first four of Chen's rules, which state that such sentences should contain common nouns, transitive verbs, adjectives and adverbs.

An Extended Entity-Relationship (EE-R) diagram that shows the full schema of the Books Recommendation DBMS. We use Elmasri's notation because it is popular in the academic community, complete, and simple to apply.

SQL code that implements the database. SQL was chosen because it is widely used and simple.

This part can be reused by researchers, instructors and developers working on similar systems.

In short, this paper focuses on the database side of a book recommendation system and covers four subtopics: general information about books, data requirements of such software, the EE-R diagram, and the SQL code that implements it.

2. Product Perspective and Related Work

2.1 Books in the Student Environment: Preferences, Reading Activity and Impact on Academic Performance

Talanov [2] looks at how young readers form their preferences. The reader's background plays a key role in reading frequency and in the choice of authors and genres. One important factor is family income: higher income tends to lead to more reading. Gender is another factor — male readers tend to choose books on politics, history and economy, while female readers tend to prefer romances, novels and classics. A third factor is the influence of friends, parents and grandparents who suggest specific books or genres. From this, we decided that the reader entity in our DBMS should include these three attributes so that recommendations can be more precise.

2.2 Artificial Intelligence Algorithms for Collaborative Book Recommender Systems

Tegetmeier et al. [5] study the use of AI algorithms inside Collaborative Filtering (CF) for book recommendation. The paper compares Memory-Based and Model-Based algorithms with two control algorithms (Popular and Random). The Model-Based method is Matrix Factorization (MF) trained with Stochastic Gradient Descent (SGD), which tries to find hidden patterns in user preferences using a latent feature space. The Memory-Based method is Book-Based k-Nearest Neighbors (kNN), which suggests books by similarity and fills missing values in the User–Book matrix.

The evaluation uses prediction metrics (RMSE, MSE, MAE) and classification metrics (Precision, Recall). MF-SGD beats kNN in every setting. Surprisingly, the Popular control gets higher Precision and Recall than both AI algorithms, and kNN cannot beat either control on prediction. The paper is a useful base for our recommendation logic, in which we plan to add Large Language Models (LLMs) and NLP for personalization.

2.3 Undergraduate Students' Reading Preference in the Extensive Reading Program: Diary Method

Prancisca et al. [6] use a diary method to study reading preferences and the factors that shape them among undergraduate students, with the goal of improving school book selection. The sample is small, but the method gives consistent and qualitative data. The top genres are Drama / Slice of Life (29.76%), Classic Tales (16.62%) and Comedy (11.8%); the least preferred are Language and Education (1.88%), Children's Book (1.88%) and Psychology (2.41%). Most participants were female, so the numbers carry some bias — among male students Classic Tales is first (38.46%) and Drama only second (17.95%). More importantly for us, the paper lists four factors that drive book choice: education level, age, gender and environment (academic and personal). These factors guide what attributes we keep in the database.

2.4 Critique of Ahmed and Letta (2023)

Ahmed and Letta [7] build a collaborative filtering recommender for university libraries to address information overload. They compare k-Nearest Neighbors with Singular Value Decomposition (SVD) on a dataset of more than 5,000 entries (ratings, book features and transactions) and conclude that SVD is more accurate, as it is less sensitive to a sparse user-item matrix and finds latent factors better. They also discuss the cold-start problem and suggest parameter tuning and regularization to help SVD on new users and books. For our DBMS, the takeaway is that matrix factorization is preferable to neighborhood-based methods. Unlike Ahmed and Letta, who look for the single best match, we aim to recommend several books, since different subtopics may live in different books. Storing extra fields like course associations, GPA and subject hierarchy gives us enough signals to soften the cold-start problem even with few ratings.

2.5 Assessment of Luekhong and Chansanam (2025)

Luekhong and Chansanam [8] compare four algorithms — Random Baseline, Popularity-based, User-Based Collaborative Filtering (UBCF) and SVD — on a large dataset (10,000+ items, 53,424 users, almost 1 million ratings). SVD reaches the lowest RMSE and is the most accurate technique, again confirming matrix factorization as a strong choice for latent user-item relations. UBCF is a middle ground between accuracy and speed. The authors also note that matrix factorization is computationally heavy, UBCF still suffers from sparsity and cold-start, and real deployments must handle time-sensitive recommendations and different domains (library vs. retail). Following these findings, our DBMS plans to use matrix factorization for accuracy, but extends the work by offering multi-book recommendations at the subtopic or chapter level and by using domain metadata (course associations, chapter indexing, topics) to fight cold-start when ratings are few.

3. Data Specification and Sources

Even though both the amount of available data and the ways to acquire it have changed a lot in the last fifty years, the data modeling process still has three steps [9]: (1) Requirements Collection — gathering data needs from interviews, surveys and questionnaires; (2) Requirements Definition — writing those needs into a Technical or Business Requirements document; (3) Requirements Visualization — drawing relationships, entities and constraints using techniques such as Entity-Relationship modeling. This section deals with the specifications produced by the first two steps [11].

3.1 Logical Requirements

Below are the most important business rules, written as Subject [Entity] – Verb [Relationship] – Object [Entity]:

Every book must belong to at least one genre.

Genres can have multiple books dedicated to them.

Every book must be written by one or more authors.

Authors could have written many books.

A person must be an author or a reader.

Readers can review many books, and any book can be reviewed by several readers.

Readers can like some books.

Books can be liked by many readers.

Books can belong to some subjects.

Subjects can have multiple books dedicated to them.

Readers can like multiple authors.

A single author can be liked by multiple readers.

Every subject must be related to one and only one genre.

Some genres may have some related subjects.

Topic aggregates subject and genre.

Readers can be interested in many topics.

3.2 Specialization Tables

Specialization (or definition) tables help when building ER and EER diagrams. They map both relationship and entity sets in detail, which leads to a cleaner final design.

3.2.1 Entity Sets

Six entity sets are defined: Book (ISBN, description, cover_image, publish_date); Person (ID, f_name, m_initial, l_name); Reader (gender, year_of_study, major_dept, years_reading, b_date); Author (nickname, num_books — derived, main_lang); Genre (ID, name, description, num_books — derived); and Subject (ID, name, description, num_books — derived). All are strong entity sets. Reader and Author specialize Person.

3.2.2 Relationship Sets

The relationship-set table covers, for every relationship, its meaning and type, the participating entities with their (min, max) constraints, and any descriptive attributes. Most of our relationships are non-identifying. Two IS-A relationships connect Person to Author and to Reader, with (1,1) on the subclass side and (0,1) on the superclass side. The remaining relationships are: belongs_to (Book–Genre, 1..* / 0..*); read (Reader–Book, 0..* / 0..*); like (Reader–Book, 0..* / 0..*); belongs_to (Book–Subject); like (Reader–Author, 0..* / 1..*); belongs_to (Subject–Genre, 1..1 / 0..*); writes (Author–Book, 0..* / 1..*); and review (Reader–Book, 0..* / 0..*). None of them carry descriptive attributes.

4. Functional Requirements

This section lists what the Books Recommendation DBMS should be able to do in terms of queries and recommendations. Items 1–6 can be handled with SQL alone, while items 7–10 need AI logic or matrix factorization on top of standard SQL.

Retrieve books for a student based on enrolled courses or GPA.

Find the top 5 most loaned books in a subject area (e.g., Physics).

Return the average rating of every book in a department (e.g., CS, Logistics).

Find students with overdue books and how long they are overdue.

Identify which authors have the most cited textbooks with many high ratings.

Find which books should be recommended based on previously loaned books with similar tags or genres.

Predict which books will be high or low rated using matrix factorization.

Create a reading list for first-year students based on collaborative filtering.

Discover latent features that make a book a bestseller for Uzbek learners using Stochastic Gradient Descent (SGD).

Recommend multiple titles on the same subject (e.g., "Database Normalization") based on subject information and prior reader feedback.

5. Data Design

As noted in Section 3, schema design is the final step of the data modeling process. In our project, this step turns the logical requirements and business rules into a complete database schema. Before writing any DBMS-specific code (MySQL, PostgreSQL, etc.), we build two models: a Conceptual Data Model that visualizes the requirements, and a Logical Data Model that

serves as the blueprint for the physical schema. The physical schema is then created using SQL Data Definition Language (DDL) [9].

5.1 Conceptual Data Design

Conceptual modeling is one way to draw the relationships and concepts of an optimized database. Skipping this step is risky and often leads to a failed project. Compared with alternatives such as Normalization, conceptual design fits best when a new database is being created, which is our case. We visualized our logical requirements with an ER diagram for most relationship sets and an Extended ER (EER) diagram for the superclass/subclass structure linking Person with Reader and Author [3, 12, 13]. The conceptual EER diagram is shown as Figure 1 in the Figures section.

5.2 Logical Data Design

Logical design turns the elements of the ER and EER diagrams — entities, aggregations and relationships — into Codd's relational model [14], which is still independent of any DBMS. We use the seven-step algorithm and its EER extension from Chapter 9 of Elmasri's Fundamentals of Database Systems [4], and apply three mapping approaches:

Foreign Key approach — for 1:M relationships and multivalued attributes.

Cross Reference (relationship-relation) approach — for M:N relationships, which cannot be represented by foreign keys alone in Codd's model.

Multiple relations / subclass relations only — for the superclass/subclass structure, since the relationship is disjoint and the subclasses have total participation.

The logical design diagram is shown as Figure 2.

5.3 Physical Data Design via MySQL Workbench

Physical design turns the logical relations into the actual low-level structures managed by the DBMS engine. Where conceptual design says what data items exist and logical design says how they relate in the relational model, physical design says where and how those relations sit on disk so that the system meets its performance and maintenance goals [1, 4]. After finishing the relational model, we exported the schema to standard SQL DDL commands (CREATE TABLE, PRIMARY KEY, FOREIGN KEY, and so on). The script fixes the data type of every attribute (ISBN as VARCHAR(17), dates as DATE, descriptive fields as TEXT), declares the primary keys, and enforces the foreign-key relationships defined in the EER diagram. Compiling the script in MySQL catches name mismatches, type problems and missing constraints — a useful first check on referential integrity before any data is loaded. We then opened the schema in MySQL Workbench and used the Reverse Engineer wizard to produce a physical diagram in Crow's-Foot notation, then tuned the participation constraints in the GUI. These changes do not touch the underlying storage but keep the diagram consistent with our ER diagram. The physical diagram is shown as Figure 3.

6. System Architecture (Block Diagram)

The block diagram describes the full user-interaction cycle of the Book Recommendation System (BookRec) — from registration to receiving personalized suggestions, while the user profile is enriched along the way through likes, reviews and chosen topics.

6.1 General Block Diagram

The general block diagram (Figure 4) starts from the Home Page. BookRec is split into four sections: Recs, Browse Books, Browse Authors and Reader's Profile.

Browse Books — list of all books, with search by title or author and a per-book page (details, reviews, like/unlike, and adding a review with rating 0–5 plus an optional comment).

Browse Authors — list of all authors with search by name and an author profile (main language, nicknames and similar metadata).

Reader's Profile — name, age, major department and reading experience, plus sub-views for Liked Books (by ISBN), Reviews (ISBN, rating, comment, date) and Liked Authors.

Recs — the recommendation logic. The user picks what to share with the AI, the pipeline runs, and the top 5 suggested books are shown.

6.2 Block Diagram for AI Implementation

The AI part runs on the server and starts from the reader's sharing preferences (Figure 5). The pipeline has these stages:

Get Sharing Preferences — read from the client what the reader allows to use from the database.

Get Reader's Data — using the reader's ID, fetch the data via SQL (MySQL in our case) and pull cover images from Amazon when needed.

Prepare Data for LLM — turn raw data into embeddings, summarize long entries, and convert structured fields into a shorter, more natural form.

Construct the Prompt — insert the prepared input into one of the prompt templates and force the LLM to return a JSON list of book ISBNs.

Gemini API Call — send the request to the model. We use Gemini 1.5 Flash for its free tier; the hyper-parameters (Top K, Top P) can be tuned beforehand.

Post-Processing — for each ISBN returned, fetch the matching row from the database and build the final JSON payload.

Return Results — send the formatted result back to the client.

7. Experiments and Analysis

This section shows that the Books Recommendation DBMS can already answer the five most important questions a reader is likely to ask. For each query we give the question in plain English and a short description of the SQL.

7.1 Books Matching the Reader's Subject or Genre Interests

Plain English: return books whose subject or genre matches the reader's major, year or topic interests (Feature 6). The query joins Reader, Readers_interested_Topics, Subject_has_Books and Book on the subject side, then unions the same join on the genre side, filtering by the reader's P_id.

7.2 Average Rating per Book in a Subject or Genre

Plain English: compute AVG(review.rating) for each book in a chosen subject or genre (Feature 3). Two parallel queries — one through Subject_has_Books and one through Genre_has_Books — join Book and is_reviewed, group by ISBN and title, and sort by average rating descending.

7.3 Authors Ranked by Total Likes

Plain English: rank authors by the total number of likes on their titles (Feature 2). The query joins Author with is_written and readers_liked_books, groups by author and orders by total_likes descending.

7.4 Books by Authors the Reader Likes

Plain English: list every book written by an author the reader has liked (Feature 2). The query joins readers_liked_authors with is_written and Book, filtered by the reader's P_id.

7.5 Classifying Books as High- or Low-Rated

Plain English: tag books that have at least ten reviews as High-Rated (average ≥ 4.0), Low-Rated (average ≤ 2.5) or Neutral (Feature 5). A CASE expression assigns the class, and the HAVING clause keeps only books with at least ten reviews and an average outside the neutral band.

8. Conclusion

In this paper we designed a Book Recommendation Database Management System for academic settings such as university libraries and student portals. The work targets a clear gap in existing recommenders: they rarely return multiple books and they rarely deal with academic subtopics. We began with an analysis of student reading preferences and existing recommendation techniques, then built the conceptual model covering books, genres, authors, subjects and user interactions. After that came the logical and physical implementations, including the SQL code and the MySQL Workbench diagram, all consistent with the conceptual model. We also reviewed collaborative filtering and matrix factorization in the literature and mapped these ideas into our functional requirements and recommendation logic, which prepares the ground for more advanced AI features later. Because the system uses GPA, academic year and reading preferences, it can give better suggestions than a popularity-only or bestseller-only baseline.

9. Open Issues and Future Work

The current system gives a solid base, but several features are still left for future work:

Tuning for specific user segments — further research can find the metrics that matter for groups such as Uzbekistani students, and the database can then be tuned to those metrics.

AI and Machine Learning models — real-time collaborative filtering algorithms such as SVD, SGD and deep-learning recommenders can lift personalization accuracy.

NLP for metadata extraction — many book records have no standardized metadata, but NLP can pull themes, subtopics and academic relevance from descriptions, which improves multi-book recommendations.

Feedback-driven learning — using ratings, likes, comments and borrow history in a learning loop allows the system to adapt and to handle cold-start better for new users and books.

Working in these directions, we aim to keep improving the quality of academic book recommendations and to keep the software modular enough for other institutions and learning platforms.

Figures

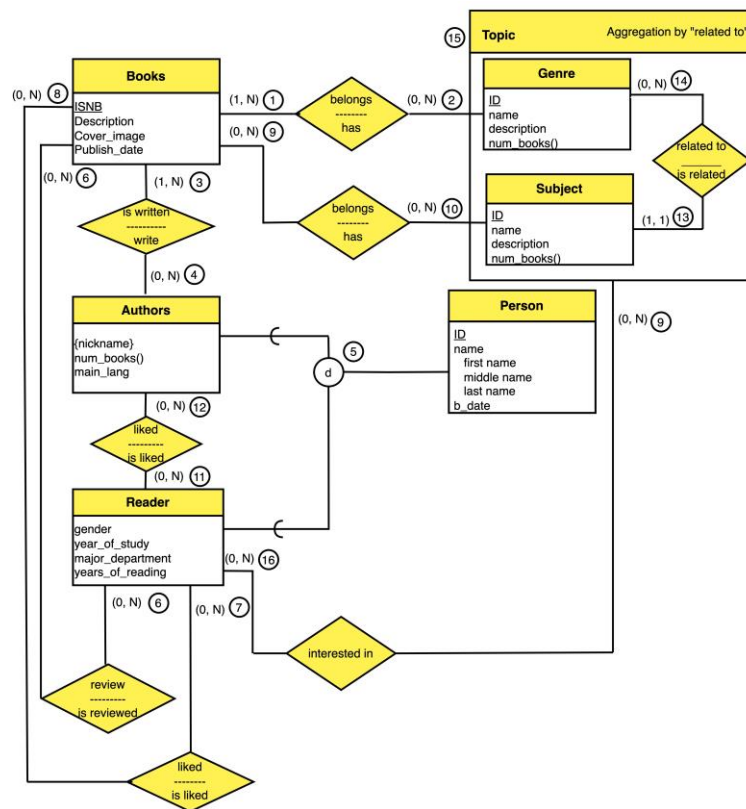


Figure 1. Conceptual EE-R Diagram.

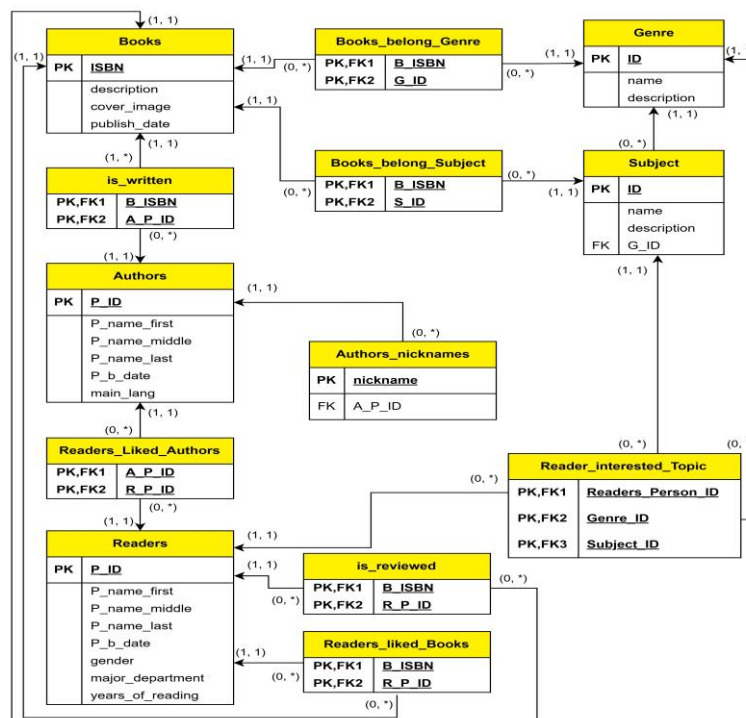


Figure 2. Logical (Relational) Diagram

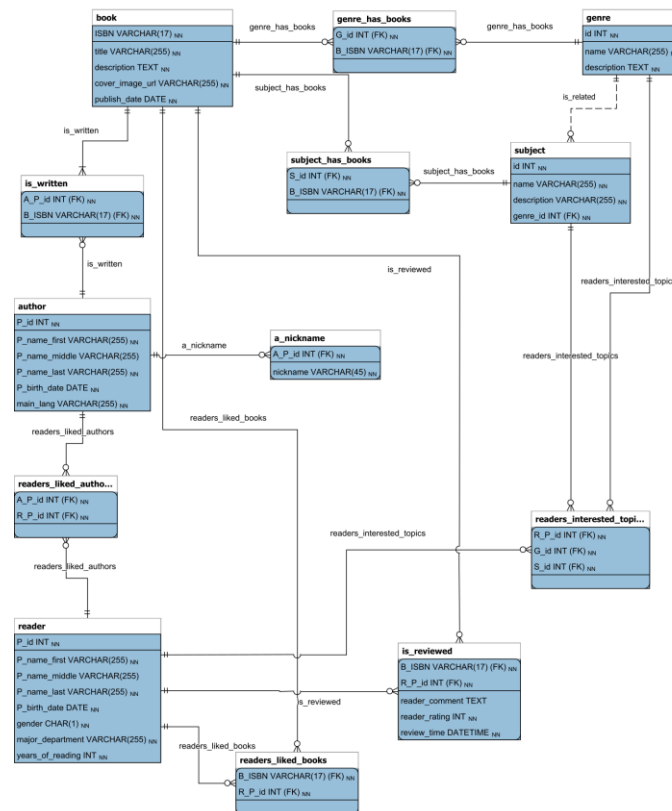


Figure 3. Physical Diagram (MySQL Workbench, Crow's-Foot notation).

References

- [1] Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. McGraw-Hill Education, 2019. ISBN 978-1-260-08706-1. URL: <https://www.db-book.com/>.
- [2] Talanov S. Books in the Student Environment: Preferences, Reading Activity and Impact on Academic Performance. Alma mater. Vestnik Vysshey Shkoly, 2023, pp. 26–41. DOI: 10.20339/AM.02-23.026.
- [3] Chen P. P.-S. The Entity-Relationship Model — Toward a Unified View of Data. ACM Transactions on Database Systems, 1976, vol. 1, no. 1, pp. 9–36. DOI: 10.1145/320434.320440.
- [4] Elmasri R., Navathe S. B. Fundamentals of Database Systems. 6th ed. Addison-Wesley, 2010. ISBN 978-0136086208.
- [5] Tegetmeier C., Johannssen A., Chukhrova N. Artificial Intelligence Algorithms for Collaborative Book Recommender Systems. Annals of Data Science, 2023, vol. 11, no. 5, pp. 1705–1739. DOI: 10.1007/s40745-023-00474-4.
- [6] Francisca S., Fergina A., Deping Z., Ramadhan I., Rizqi M. A., Ananda E. Undergraduate Students' Reading Preference in the Extensive Reading Program: A Diary Method. Journal of English Language Teaching Innovations and Materials (JELTIM), 2023, vol. 5, no. 2, pp. 110–124. DOI: 10.26418/jeltim.v1i1.67998.
- [7] Ahmed E., Letta A. Book Recommendation Using Collaborative Filtering Algorithm. Applied Computational Intelligence and Soft Computing, 2023, vol. 2023, pp. 1–12.

- 
- [8] Luekhong P., Chansanam W. Advancing Book Recommendation Systems: A Comparative Analysis of Collaborative Filtering and Matrix Factorization Algorithms. *Journal of Information Systems Engineering and Management*, 2025, vol. 10, no. 4s, pp. 1–12.
- [9] Sharma A. Examining the Current Utilization of Conceptual Data Modeling in Database Modeling Practices: Mixed Methods Research. PhD dissertation, Loyola University of Chicago Graduate School, 2025. URL: https://ecommons.luc.edu/luc_diss/4151.
- [10] IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications. IEEE, 1998, pp. 1–40. DOI: 10.1109/IEEESTD.1998.88286.
- [11] de Weck O. L. Fundamentals of Systems Engineering — Session 2: Requirements Definition. MIT OpenCourseWare, 2015.
- [12] Thalheim B. The Enhanced Entity-Relationship Model. In: Embley D. W., Thalheim B. (eds.) *Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges*. Springer, Berlin, Heidelberg, 2011, pp. 165–206. DOI: 10.1007/978-3-642-15865-0_6.
- [13] Bagui S. Mapping OWL to the Entity Relationship and Extended Entity Relationship Models. *International Journal of Knowledge and Web Intelligence*, 2009, vol. 1, no. 1–2, pp. 125–149. DOI: 10.1504/IJKWI.2009.027929.
- [14] Codd E. F. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 1970, vol. 13, no. 6, pp. 377–387. DOI: 10.1145/362384.362685.