RV College of Engineering®, Bengaluru – 59
Department of Computer Science and Engineering
DATABASE MANAGEMENT SYSTEMS (CD252IA)

Synopsis

TITLE: CiteMind: A Hybrid Knowledge Graph for Research Paper Retrieval		
TEAM	USN:	Name
	1RV23CS155	Nikita S Raj Kapini
	1RV23CS161	Nitya P Mohare
	1RV23CS177	Prarthana P Kulkarni

With millions of papers published annually, researchers struggle to efficiently find relevant work, understand citation intent, and track emerging trends. Traditional databases like Google Scholar and arXiv rely on keyword-based search, missing semantic context and deeper relationships. CiteMind overcomes these limitations using a hybrid system—SQL, NoSQL, and Knowledge Graphs, enhanced by NLP to enable semantic search, citation analysis, and research network visualization.

Existing System
Current repositories store structured metadata and depend on keyword matching, overlooking semantic similarity and citation context. Citation analysis is limited to counts, and author-topic-paper relationships are poorly modeled. Unstructured data like abstracts remain underutilized, forcing researchers to manually explore literature and collaborations.

Proposed System

1. **SQL (Relational Database):** Stores structured metadata such as paper titles, authors, journals, and citation links.
2. **NoSQL (Document Database):** Stores unstructured data including abstracts, NLP outputs, semantic embeddings, and topic models.
3. **Knowledge Graph:** Represents authors, papers, topics, and citations as nodes and edges for advanced relationship reasoning.

Key Features:

- **Semantic search:** Retrieve papers based on meaning rather than keywords.
- **Citation context analysis:** Classify why a paper cites another (support, contrast, extension).
- **Topic trend detection:** Analyze the evolution of research topics over time.
- **Author influence analysis:** Identify key authors using graph metrics like centrality and

PageRank.

- **Visualization:** Display co-author networks, citation graphs, and topic clusters.

Novelty: Unlike traditional systems, CiteMind combines structured, unstructured, and relational data under a unified, intelligent framework, enabling **context-aware, semantic research exploration**.

Relational Database Structure

The **SQL layer** stores structured data following a relational model:

Table	Attributes	Description
Authors	author_id (PK), first_name, last_name	Metadata about researchers
Papers	paper_id (PK), submitter, title, year, journal_id (FK), abstract_id (FK), comments, journal_ref, doi, citation_count, category, license, report_no	Main bibliographic information
Journals	journal_id (PK), name, publisher	Journals and publishers
Author_Paper	author_id (FK), paper_id (FK)	Many-to-many mapping between authors and papers
Versions	version_id(PK),paper_id(FK), version, created	Version history of listed papers

Relationships:

- One paper may have multiple authors.
- One paper may cite multiple other papers.
- Authors can contribute to multiple papers.

This structure allows efficient metadata queries such as “list all papers by a given author” or “retrieve the top-cited papers in a journal.”

RDBMS AND NoSQL Integration

The **NoSQL layer (MongoDB)** manages unstructured and semi-structured data that cannot be efficiently stored in SQL tables. Collections example includes abstracts, NLP outputs which includes keywords, topic models, semantic embeddings and summaries and citation contexts that explains why a paper cites another.

Integration Approach:

- SQL paper IDs act as common keys linking structured metadata with unstructured content. Hybrid queries combine **relational joins** (e.g., author-paper mapping) with **semantic search** in NoSQL using embeddings.
- Knowledge Graph nodes reference both SQL and NoSQL data for enriched visualization and analytics. This integration enables **context-aware retrieval**, combining metadata, full-text understanding, and relationship reasoning.

The user of the application can create a profile and navigate through papers. Additionally, NoSQL components will be used to support note making for each paper being referred by the user. The admin of the application will control addition of papers to the repository.

Societal Concern

CiteMind addresses the challenge of information overload in academic research by enabling researchers, students, and institutions to access relevant literature efficiently, identify influential authors and collaborations, detect emerging trends and gaps in research, and reduce duplication of effort in scholarly work. The system leverages several technological trends, including Natural Language Processing (NLP) for keyword extraction, topic modeling, embeddings, and summarization; Knowledge Graphs for semantic relationship modeling; hybrid SQL–NoSQL integration for unified data management; and graph analytics, such as centrality and PageRank, to evaluate author and citation influence. By combining AI with hybrid database technologies, CiteMind contributes to intelligent knowledge management, supporting innovation, enhancing collaboration, and improving the overall efficiency of scholarly research.

Software Requirement specification

A software requirements specification (SRS) is a description of a software system to be developed. The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.

Hardware Requirements

- **Processor:** Quad-core CPU (Intel i5 / AMD Ryzen 5 or equivalent)
- **RAM:** 8 GB
- **Storage:** 20 GB available disk space
- **Network:** Stable broadband connection for data ingestion and querying
- **GPU:** Since NLP model fine-tuning is performed

Software Requirements

- **Operating System**
  - Windows 10/11, Linux (Ubuntu 20.04+), or macOS
- **Database Software**
  - MySQL – relational database for structured metadata
  - MongoDB – NoSQL document store for abstracts, embeddings, and NLP outputs
  - FireStore – NoSQL document store for managing user details and data associated with each user
- **Application Frameworks**
  - Backend: Node.js with Express.js
  - Frontend: React Typescript
- **NLP and ML Libraries**
  - Transformers (HuggingFace)
  - NLTK
  - Scikit-learn
  - Sentence-BERT / OpenAI embeddings
- **Visualization Tools**
  - D3.js, Plotly
- **Other Tools**
  - GitHub for version control
  - Postman for API testing

Functional Requirements

1) Research Paper Ingestion and Metadata Management

- The system shall allow ingestion of research papers (PDF/metadata) into the SQL database.
- The system shall extract structured fields such as *title, authors, year, journal,* and citation links.
- The system shall maintain relational integrity across **Authors**, **Papers**, **Journals**, and **Citations** tables.

2) NLP Processing and Semantic Embedding Generation

- The system shall store unstructured content (abstracts, summaries, keywords) in MongoDB.
- The system shall generate semantic embeddings using pre-trained NLP models.
- The system shall perform topic modeling, keyword extraction, and citation intent classification.
- The system shall connect SQL metadata with NoSQL NLP outputs through unique paper IDs.

3) Knowledge Graph Construction and Graph Analytics

- The system shall construct a knowledge graph in using nodes for authors, papers, topics, and citation relationships.
- The system shall support graph queries such as:
  - finding co-author networks
  - discovering citation paths between papers
- The system shall synchronize graph nodes with SQL and NoSQL data.

4) Semantic Search and Intelligent Retrieval

- The system shall allow users to search papers using natural language queries (semantic search).
- The system shall retrieve results ranked by embedding similarity, citation strength, and topic relevance.
- The system shall support hybrid queries combining SQL filters (e.g., year, journal) and semantic NoSQL search.
- The system shall provide advanced filtering features such as:
  - topic-based exploration
  - citation intent filtering (support/contrast/extension)

5) User and Admin accounts

- There will be two kinds of users— general users and admins.
- General users will be able to make notes and mark the papers they are referring to and contribute to the citation context being established
- Admins will be responsible for updating the repository with new papers

Table-Wise Normalization Analysis

1. PAPERS TABLE
Schema:
papers(paper_id, submitter, title, comments, journal_ref, doi, report_no, license, update_date, selected_category, citation_count)

First Normal Form (1NF):
All attributes in the papers table store atomic values such as strings, integers, or dates. There are no repeating groups or multi-valued attributes. Each row uniquely represents one research paper using the primary key paper_id. Hence, the table satisfies First Normal Form.

Second Normal Form (2NF):
The primary key of the papers table is a single attribute, paper_id. All non-key attributes such as title, submitter, and citation_count are fully functionally dependent on this key. Since there is no composite key, partial dependency cannot occur. Therefore, the table satisfies Second Normal Form.

Third Normal Form (3NF):
There are no transitive dependencies in the papers table. No non-key attribute depends on another non-key attribute; instead, all attributes depend directly on paper_id. Thus, the table satisfies Third Normal Form.

Boyce–Codd Normal Form (BCNF):
All functional dependencies in this table have paper_id as the determinant, and paper_id is a superkey. Hence, the papers table satisfies BCNF.

2. AUTHORS TABLE
Schema:
authors(author_id, first_name, last_name)

First Normal Form (1NF):
Each attribute in the authors table contains atomic values. There are no lists, arrays, or composite fields. Each row represents exactly one author. Hence, the table satisfies First Normal Form.

Second Normal Form (2NF):
The primary key is author_id, a single attribute. All non-key attributes (first_name and last_name) depend entirely on author_id. Therefore, the table satisfies Second Normal Form.

Third Normal Form (3NF):
There are no transitive dependencies in the authors table. Author names do not determine any other non-key attributes. Hence, the table satisfies Third Normal Form.

Boyce–Codd Normal Form (BCNF):
The only determinant in the table is author_id, which is a superkey. Therefore, the authors table satisfies BCNF.

3. PAPER_AUTHORS TABLE
Schema:
paper_authors(paper_id, author_id)

First Normal Form (1NF):
Each row in the paper_authors table represents a single association between a paper and an author. There are no repeating groups or multi-valued attributes. Hence, the table satisfies First Normal Form.

Second Normal Form (2NF):
The table uses a composite primary key consisting of paper_id and author_id. There are no non-key attributes in this table, so partial dependency is not possible. Therefore, the table satisfies Second Normal Form.

Third Normal Form (3NF):
Since there are no non-key attributes, transitive dependency does not exist. The table trivially satisfies Third Normal Form.

Boyce–Codd Normal Form (BCNF):
The composite primary key uniquely identifies each row, and there are no other functional dependencies. Hence, the paper_authors table satisfies BCNF.

4. VERSIONS TABLE
Schema:
versions(version_id, paper_id, version, created)
with a UNIQUE constraint on (paper_id, version)

First Normal Form (1NF):
All attributes in the versions table contain atomic values. Each row represents a single version of a paper, and there are no nested or multi-valued attributes. Thus, the table satisfies First Normal Form.

Second Normal Form (2NF):
The primary key is version_id, and all other attributes (paper_id, version, created) are fully functionally dependent on this key. There are no partial dependencies. Hence, the table satisfies Second Normal Form.

Third Normal Form (3NF):
There are no transitive dependencies among non-key attributes. Version-related information is stored independently of paper metadata. Therefore, the table satisfies Third Normal Form.

Boyce–Codd Normal Form (BCNF):
From the semantics of the data, the creation date of a version depends on the combination of paper_id and version. This functional dependency is enforced using a UNIQUE constraint on (paper_id, version), making it a candidate key. As a result, every determinant is a superkey, and the table satisfies BCNF.

FINAL CONCLUSION

Each table in the schema was derived by decomposing the original JSON document based on functional dependencies and relationship cardinalities. Atomic attributes ensure First Normal Form, elimination of partial dependencies ensures Second Normal Form, and removal of transitive dependencies ensures Third Normal Form. Additional constraints based on real-world semantics allow the schema to satisfy Boyce–Codd Normal Form. The final design is free from redundancy and update anomalies and accurately represents the structure of the original JSON data in a relational form.

Design of Forms, Security and Validation

1. Design of Forms

The CiteMind system employs modern, responsive, and user-centric form design to ensure efficient interaction for both general users and administrators. The frontend is developed using React with TypeScript, enabling component-based architecture, strong type safety, and scalable UI design.Key Forms in CiteMind

a) User Registration and Login Forms
User registration is enabled through google accounts. Admin login enforces role-based access control (RBAC) at the authentication level. Admin ids are separately handled. These forms support real-time field validation, password strength indicators, and inline error feedback, ensuring a smooth and secure user experience. Authentication is handled securely using Firebase Authentication.

b) Research Paper Search Form
The search form supports natural language input for semantic search and includes advanced filters such as publication year, journal, topic, and citation intent (support/contrast/extension). It is implemented using controlled React components with debounced input handling to optimize performance and reduce unnecessary backend requests.

c) Paper Detail and Note-Making Form
This form allows users to add personal notes, bookmark papers, and mark them as referred. Notes are stored in Firestore NoSQL, linked to both user ID and paper ID, ensuring personalization and data isolation. Rich-text support enables structured academic note-taking.

d) Admin Paper Ingestion Form
The admin ingestion form allows uploading of metadata and/or PDF files. Fields include Title, Authors, Journal, Year, DOI, and Abstract (manual or extracted). The form integrates with backend APIs for SQL insertion and triggers MongoDB-based NLP pipelines for further processing.

All forms are modular, reusable, and styled using modern UI libraries to ensure consistency, scalability, and maintainability.

2. Security Design

Security in CiteMind is implemented across frontend, backend, and database layers, ensuring confidentiality, integrity, and controlled access.

a) Authentication and Authorization
Firebase Authentication is used for secure identity management. Role-based access control differentiates general users (search, view, annotate) and admins (add/update papers, manage repository). JSON Web Tokens (JWT) are used to authorize API requests.

d) Database Security
The SQL database uses parameterized queries to prevent SQL injection and foreign key constraints to maintain relational integrity. MongoDB and Firestore restrict access through authenticated user context, and Firestore security rules ensure user notes remain isolated.

e) Input Sanitization and Protection
User inputs are sanitized to prevent XSS and injection attacks. Admin file uploads are validated for file type and size to block malicious payloads.

3. Validation Mechanisms

CiteMind implements multi-layer validation to ensure data accuracy and robustness.

a) Frontend Validation
Frontend validation is implemented using React Hook Form and TypeScript, enforcing required fields, email format validation, password complexity rules, and search query length constraints. Immediate inline feedback enhances user experience.

b) Backend Validation
Express.js middleware validates incoming requests using schema validation libraries ensuring mandatory metadata presence, correct data types for SQL/NoSQL storage, and valid paper ID references.

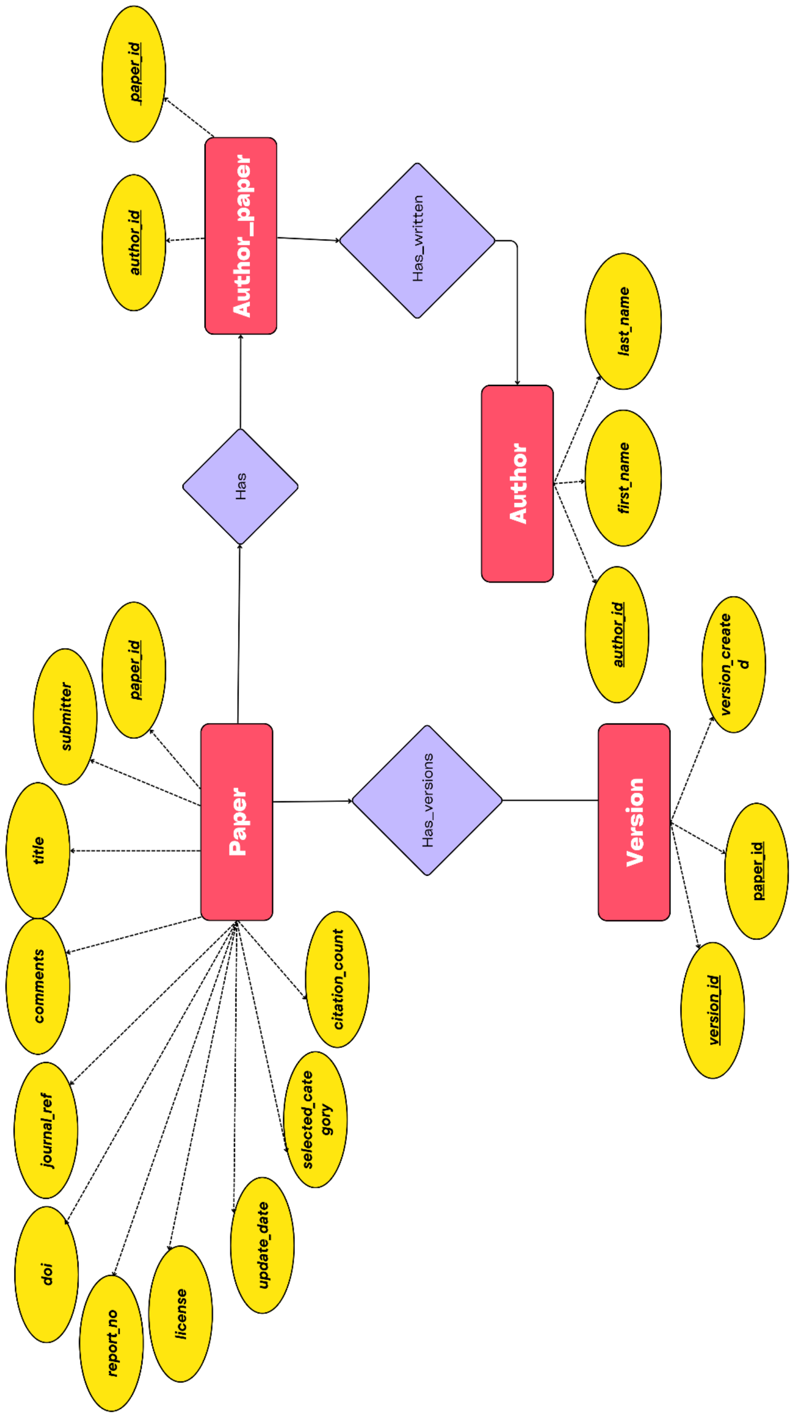
c) Database-Level Validation
SQL enforces primary keys, foreign keys, and unique constraints. NoSQL consistency is maintained through application-level validation, while Firestore rules ensure users can only modify their own notes and bookmarks.

4. Use of Modern Engineering Tools

CiteMind demonstrates effective use of modern engineering tools. React with TypeScript enables scalable frontend development, while Firebase Authentication and Firestore provide secure user management and personalized data storage. Express.js middleware centralizes security and validation logic. NLP pipelines and knowledge graph integration are securely triggered via backend APIs. GitHub supports version control and collaboration, and Postman is used for API testing and security validation.

5. Summary

The design of forms, security, and validation in CiteMind ensures a secure, scalable, and user-friendly research exploration platform. By integrating modern frontend frameworks, secure authentication mechanisms, layered validation strategies, and role-based access control, the system fulfills both functional and non-functional requirements. This component effectively demonstrates the application of modern engineering tools and best practices for intelligent, semantic-driven academic research.



ER DIAGRAM AND SCHEMA

USERS

ID	Username	Email

ADMINS

ID	Username	Email

PAPER_AUTHORS

Author_ID	Paper_ID

AUTHORS

Author_ID	First_name	Last_name

PAPERS

_id	Submitter	Title	Comments	journel_ref	doi	report_no	license	updated_date	selected_category	citation_count

VERSIONS

Version_id	Version	Created	Paper_id

VECTOR_EMBEDDINGS

Summary	Keywords	Fields	Novelty	Related_work	_id

NLP_OUTPUT

Paper_id	_id

CITATION_CONTENT

Citing	Cited	_id	Context	Label

ABSTRACT

Paper_id	_id	Abstract

PAPER_METADATA

_id	Paper_id	Title	Authors	raw_authors_string	Categories	selected_categorgy	Citation_Count	Year