AI LitReview: A Web-Based Tool for LLM-Assisted Systematic Literature Reviews

Anderson Deizepe
Universidade Federal do Paraná - UFPR
Curitiba, Brasil
anderson@deizepe.com.br

Katia Romero Felizardo
Universidade Tecnol. Federal do Paraná - UTFPR
Cornélio Procópio, Brasil
katiascannavino@utfpr.edu.br

Roberto Pereira
Universidade Federal do Paraná - UFPR
Curitiba, Brasil
rpereira@inf.ufpr.br

Abstract

Conducting Systematic Literature Reviews (SLRs) is an essential but labor-intensive activity in Software Engineering (SE) research. Existing tools offer limited support for AI-assisted analysis and lack features such as local model execution, multi-user collaboration, templates, explainability, and integrated cost tracking. This paper presents AI LitReview, a collaborative web-based tool that supports the entire SLR workflow, from protocol definition to report generation, with integrated support for Large Language Models (LLMs) via OpenRouter API and locally via Ollama. The tool enables multiple activities, such as: protocol elaboration during the planning phase, as well as initial selection, final selection, data extraction, and report during the execution phase. Each activity records token usage, processing time, and cost per study, promoting transparency and reproducibility. A Sankey-based Study Flow Diagram and XLS export at each phase further support auditability. A Retrieval-Augmented Generation (RAG) module synthesizes answers to research questions from the included studies. AI LitReview is available at https://ailitreview.dzp.net.br.
Demo video: https://zenodo.org/records/20387266

Keywords

Systematic Literature Review, SLR, Large Language Models, LLM, AI-Assisted tool, Software Engineering, Support Tool

1 Introduction

Systematic Literature Reviews (SLRs) have become an essential method in Software Engineering (SE) for synthesizing evidence of primary research, supporting decisions about what is known and what remains unexplored [8, 13]. Despite their value, SLRs impose a substantial burden on researchers, the process encompasses protocol definition, multi-searching strategies, multi-phase screening, data extraction, and synthesis, all requiring methodological rigor and extensive manual effort [8]. Studies have consistently highlighted that SLR execution is time-consuming and error-prone and that documentation and quality assessment practices still present important shortcomings [7, 12].

Tool support has been recognized as indispensable to make SLRs tractable on scale [17]. Existing platforms such as StArt [5], Parsifal [10], and Thoth [4] address important needs, protocol management, study organization, and collaborative workflows, but were designed before the widespread availability of Large Language Models (LLMs) and provide limited or no support for AI-assisted analytical tasks. The study selection activity is particularly demanding; it often involves manually screening hundreds or thousands of candidate studies based on title, abstract, and keywords, an activity that consumes a substantial portion of the total SLR effort [2, 13].

Recent empirical work has shown that LLMs can support SLR study selection with an accuracy ranging from 75% to 86%, depending on the model and the review context [7]. These results indicate that LLMs are not accurate enough to replace human judgment but can serve as a valuable supplementary reviewer. Other limitations include: [6]: (i) LLMs are sensitive to prompt wording; therefore, researchers need to experiment with different prompt templates across distinct SLR activities, such as study selection and data extraction, and compare the results against human-generated results (oracle).; (ii) Due to the non-deterministic nature of LLM outputs, exact model versions, parameters, prompts, and execution metadata should be documented for reproducibility; (iii) running multiple screenings in large corpora with proprietary models incurs significant costs that must be tracked[11]; (iv) the final selection activity, reading full texts, demands access to PDFs and larger context windows; and (v) data extraction and report synthesis remain underexplored activities in which LLMs show promise but lack tool support[6]. These challenges collectively call for a platform that goes beyond single-model, single-phase support.

This paper presents **AI LitReview**, a collaborative web-based platform designed to support the SLR process. The tool covers protocol definition, study import, duplicate removal, AI-assisted initial and final selection, structured data extraction, and RAG-based report generation, with configurable LLM support throughout the review process. The tool follows a human-in-the-loop design principle, i.e., LLMs act as supplementary reviewers that read, score, and justify, while every inclusion, exclusion, and extraction decision remains a human action, recorded together with the responsible researcher.

Researchers can perform multiple study selection activities using different models (either local models via Ollama or cloud-based models through the OpenRouter API, which provides access to more than 365 models) and custom prompt templates. The AI-generated outputs can then be compared with one another to support human inclusion/exclusion decisions. In addition, token usage, latency, and cost per study are recorded for each execution, enabling researchers to make evidence-based and cost-aware decisions regarding model selection and use.

The remainder of this paper is organized as follows: Section 2 presents background concepts; Section 3 discusses related work; Section 4 describes AI LitReview; Section 5 presents a usage example; and Section 6 concludes the article and outlines future work.

2 Background

Table 1 compares AI LitReview with the most relevant existing tools. P = partially supported; ✓ = fully supported; – = not supported.

An SLR in SE follows a well-defined process based on the Kitchenham and Budgen guidelines [14]. The process comprises three main phases: *planning* (defining the protocol, research questions, inclusion/exclusion criteria, and search strings), *execution* (searching, selecting studies through multiple activities, and extracting data), and *reporting* (synthesizing results and answering research questions).

LLMs have recently emerged as powerful tools for natural language understanding tasks [3]. In the SLR context, they can assist with study selection classification, structured data extraction, and evidence synthesis. However, their adoption in SE research introduces specific technical and methodological challenges [6]. Prompt sensitivity is a major concern, even minor variations in wording, context description, or output format constraints can alter a model decisions. LLM outputs are also inherently non-deterministic, i.e., the same prompt may produce different results across runs. Therefore, controlling parameters such as temperature and top-p, and recording exact model versions is essential for reproducibility. Cost is another practical barrier, as running API-based models over hundreds of studies per phase can accumulate significant expenditure, while local models mitigate this at the cost of hardware requirements. Finally, applying LLMs to the final selection stage requires processing full-text PDFs, which demands larger context windows and robust PDF handling infrastructure. Local inference via tools such as Ollama addresses cost and privacy concerns by allowing institutions to run open-source models on their own infrastructure without sending study content to external servers.

Ollama was chosen specifically because it is open source, exposes a simple HTTP API, and allows a wide range of open-weight models (e.g., Llama, Mistral, Qwen) to be run on commodity hardware. The trade-off is practical, local models are free and data is kept private, but they are typically smaller and less accurate than state-of-the-art API models, and inference speed is determined by the available hardware.

These challenges motivate the adoption of complementary approaches. RAG [15] combines retrieval of relevant document passages with LLM generation, enabling the model to produce answers grounded in a specific document corpus rather than relying solely on parametric knowledge. In the SLR context, RAG mitigates some of the aforementioned limitations: by restricting the model's input to retrieved excerpts rather than full documents, it reduces context window demands; by enabling the use of local models over a controlled corpus, it addresses both cost and privacy concerns. RAG can be applied to extract data and synthesize answers to research questions from the full-text of included studies.

We adopted RAG rather than direct long-context prompting because it grounds each answer in retrieved passages that can be traced back to specific studies, keeps token consumption bounded regardless of corpus size, and remains compatible with local models, whose context windows are smaller. RAG outputs nevertheless inherit LLM limitations, answers may omit relevant passages or contain hallucinations, and must therefore be treated as drafts to be verified by the researcher against the included studies.

3 Related Work

Marshall and Brereton [17] conducted a systematic mapping of SLR tool support, identifying tools such as SLuRp[1], StArt, and SLR-Tool[9] across different functional areas. Table 1 compares AI LitReview with the most relevant existing tools.

Parsifal [10] supports protocol definition and study selection but lacks AI integration and data extraction features. StArt [5] provides comprehensive SLR support for the Brazilian community, but is a desktop application without AI assistance. Thoth 2.0 [16] recently introduced snowballing support but does not integrate LLMs. Rayyan[18] offers collaborative abstract selection with some AI-powered duplicate detection but does not support full-text selection or data extraction.

The closest related tool to AI LitReview is AISysRev [12], a web application (React + FastAPI + PostgreSQL) for AI-assisted title-abstract screening. AISysRev was directly motivated by the LLM based-methodolgy proposed by Felizardo et al. [7].

In this methodology, instead of asking the LLM for a single include/exclude verdict, the model rates each inclusion and exclusion criterion on a 7-point Likert scale that expresses how strongly it agrees that the study satisfies the criterion (1 = strongly disagree, 4 = neutral, 7 = strongly agree). The 7-point scale is adopted, rather than the more common 3- or 5-point variants, to capture finer-grained uncertainty and support thresholding decisions on borderline studies. AISysRev extends this scheme with two alternative output formats: a binary include/exclude classification and an inclusion probability between 0.0 and 1.0.

The tool supports multiple models via OpenRouter and OpenAI's API, and parallelizes screening calls by dispatching classification tasks across multiple queue workers that process studies simultaneously, reducing the total screening time. However, AISysRev has limitations compared to AI LitReview: (i) it does not support local model execution via Ollama or any other local provider, requiring external API access for all inference; (ii) its prompt template does not offer mechanisms for users to define, customize, or experiment with different prompting strategies. In contrast, AI LitReview provides a fully configurable per-project template system with dynamic variables; and (iii) it covers exclusively the title-abstract selection stage, with no support for protocol definition, full-text PDF selection, data extraction, report generation, or multi-user collaboration.

The SESR-Eval benchmarking study [11] provides comprehensive evidence to date on the performance of LLMs in supporting study selection activities. 9 models were evaluated against 34,528 labeled studies of 24 SE secondary studies. The study found that i) the accuracy varies between 0.34 and 0.85, ii) how the review protocol defines and describes the selection criteria, particularly the degree to which they can be objectively interpreted, has a greater effect on the accuracy than the choice of LLM; and iii) no model reached the recommended recall threshold $\geq$0.95 with precision around 0.50. The SESR-Eval prompt template was built on the zero-shot prompt approach proposed by Huotala et al. and the Likert-based methodology introduced by Felizardo et al. [7], demonstrating the progressive refinement of this line of research.

A natural question is why AI LitReview was developed as a new tool rather than as an extension of open-source platforms such as StArt, Parsifal, or Thoth. Three reasons guided this decision. First,

Table 1: Comparison of SLR tools (P = partially supported)

Feature	SLuRp	StArt	SLR-Tool	Parsifal	Thoth	Rayyan	AISysRev	AI LitReview
Protocol definition	–	✓	✓	✓	✓	–	–	✓
Automated search support	–	P	–	–	–	–	–	–
Study selection	✓	✓	✓	✓	✓	✓	✓	✓
Quality assessment	✓	–	P	✓	✓	–	–	✓
Data extraction	✓	✓	✓	✓	✓	–	–	✓
Computational text analysis	✓	✓	–	–	–	–	–	✓
Meta-analysis	P	–	–	–	–	–	–	–
Report writing	P	P	P	P	–	–	–	✓
PDF bulk upload	–	–	–	–	–	–	–	✓
AI-assisted screening	–	–	–	–	–	–	✓	✓
Local LLM support (Ollama)	–	–	–	–	–	–	–	✓
Multi-model comparison	–	–	–	–	–	–	✓	✓
Configurable prompt templates	–	–	–	–	–	–	–	✓
Full-text (PDF) screening	–	–	–	–	–	–	–	✓
Cost tracking per study	–	–	–	–	–	–	–	✓
RAG-based report generation	–	–	–	–	–	–	–	✓
XLS export per phase	–	–	–	–	–	–	P	✓
Collaboration	–	–	–	✓	✓	✓	–	✓
Open source / free	–	✓	–	✓	✓	–	✓	✓

StArt is a desktop application, whereas multi-user collaboration and asynchronous LLM orchestration require a web architecture with queue workers and centralized storage. Second, the LLM-related requirements, per-study execution logs (tokens, latency, cost) in a document store, PDF handling in object storage, and provider abstraction over OpenRouter and Ollama, are cross-cutting concerns that would demand deep changes to the persistence and processing layers of the existing codebases, rather than the addition of an isolated module. Third, starting from a clean data model allowed the design of prompt templates and multiple screenings per phase as first-class concepts. Integration with existing tools is nevertheless supported as a complementary path: the per-phase XLS export allows results to flow to other tools in the researcher's pipeline.

To our knowledge, AI LitReview is the first tool to integrate multi-phase AI assistance (initial selection, full-text PDF selection (final selection), data extraction, and RAG-based synthesis) with configurable prompt templates, per-study cost, and token tracking.

4 AI LitReview

AI LitReview is a collaborative web application designed to support researchers throughout the entire SLR process, from protocol definition to evidence synthesis. The backend was built with Laravel (PHP-FPM), providing a structured MVC architecture for managing projects, users, studies, and AI integrations. Relational data, including projects, studies, screening decisions, and extraction results, is stored in MySQL, while execution logs from AI screenings (token counts, latency, cost, and raw model responses) are persisted in MongoDB, decoupling high-volume log writes from the transactional data model. Study PDFs are stored in S3-compatible object storage and retrieved on demand during full-text selection and RAG-based report generation. The application supports multi-user, multi-project (SLRs); researchers can create SLR projects, invite collaborators by email, and work concurrently on different SLR activities. Role-based access and per-SLR data isolation ensure that each team's work remains independent. The tool is publicly available at https://ailitreview.dzp.net.br and is released under the MIT license.

The following subsections detail AI LitReview's main components. The system architecture and the overall SLR process are described first, followed by each supported activity in sequence: project management and protocol definition, prompt template configuration, AI-assisted study selection (initial and final screening), structured data extraction, and RAG-based report generation. Each activity is supported by asynchronous processing, which ensures that long-running AI tasks do not block the user interface.

4.1 Architecture and SLR process

Figures 1 and 2 illustrate the system architecture and the six-phase SLR process, respectively. The *web layer* (Nginx) serves as an application for researchers. The *application layer* (Laravel/PHP-FPM) handles business logic, user management, project management, and API orchestration. Long-running AI tasks are processed asynchronously by two *queue workers*, preventing UI blocking during screening and extraction operations. Study PDFs are stored in *S3-compatible object storage*, while structured data (projects, studies, users) reside in *MySQL*. Execution logs from AI screenings, including token counts, latency, cost, and raw model responses, are persisted separately in *MongoDB*, decoupling high-volume log writes

from the relational data model. AI inference is routed either to the *OpenRouter API*, which provides unified access to over than 365 models (GPT, Claude, Gemini, DeepSeek, and others) with pay-as-you-go pricing, or to a configurable *Ollama* instance for local, cost-free inference. Users configure their OpenRouter API key and Ollama URL via the Settings interface (Figure 3).

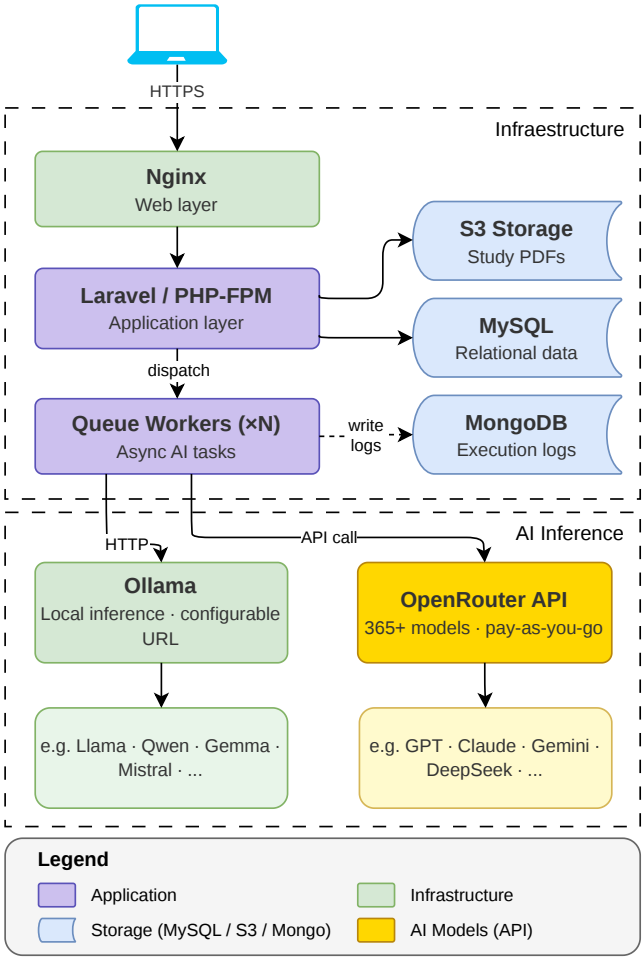

Figure 1: AI LitReview architecture: Nginx, Laravel/PHP-FPM, queue workers, MySQL (relational data), MongoDB (execution logs), S3 (PDFs), OpenRouter API, and Ollama.

4.2 Project Management and Protocol Definition

To start a new SLR, a researcher creates a project and invites collaborators by email. Each project includes a complete SLR protocol: *research questions* (RQs), *search strings* (database), *inclusion criteria* (ICs), and *exclusion criteria* (ECs) and data extraction form. The protocol is used to auto-populate AI prompt templates and structure the final report. Searches for candidate studies are performed externally to the tool; however, studies can be imported via Excel/CSV files, added manually or uploaded in bulk as PDFs documents; the system automatically assigns PDFs to studies when the filename matches the study ID (e.g. `S1.pdf`).

Regarding copyright, the PDFs are obtained by the researchers themselves through their institutional or personal access rights; the tool stores them in private, per-project object storage, uses them only as input for the analyses requested by the project members, and does not redistribute or publicly expose the files.

4.3 Prompt Templates

A key design feature is the configurable prompt template system. For example, researchers define named templates for each activity, such as study selection and data extraction, using dynamic variables that are automatically replaced at runtime with study-specific content. The variables `[[title]]`, `[[abstract]]`, and `[[fullstudy]]` inject the study metadata used in the selection activity; the first two for initial selection (title and abstract only, consistent with the SLR guidelines [13]), and the latter for final selection with full PDF text. The variables `[[inclusioncriteria]]` and `[[exclusioncriteria]]` are automatically populated in the prompt with the inclusion and exclusion criteria defined in the protocol. This allows the same template to be reused in multiple projects without manual editing.

For data extraction, `[[jsondata]]` automatically includes all data extraction fields in the prompt, instructing the model to return a structured JSON object with one value per field. Templates can encode domain-specific instructions, such as adopting the perspective of a researcher in a given field or applying the 1-7 Likert scale per inclusion criterion introduced by Felizardo et al. [7]. This flexibility allows researchers to experiment with different prompting strategies and compare results between models and runs.

These design choices operationalize findings from prior empirical work: since the wording of the selection criteria affects accuracy more than the choice of LLM [11], templates turn the criteria description into an explicit, editable experimental variable; and since no single model is reliable enough to act alone [7], the tool encourages running multiple screenings whose outputs can be contrasted with each other and with a human oracle.

Listings 1-3 show the prompts and their corresponding JSON outputs for three LLM-supported activities.

Listing 1: Initial screening: prompt input and JSON output.

```
-- INPUT --
Assume you are a software engineering researcher [...]
Title: [[title]]
Abstract: [[abstract]]
Rate your agreement (1-7 Likert) for each criterion:
[[inclusioncriteria]]
[[exclusioncriteria]]
Return ONLY valid JSON.
-- OUTPUT --
{
  "criteria": [
    { "id": "IC1", "score": 6,
      "justification": "Study focuses on..." },
  ],
  "overall": "include"
}
```

Listing 2: Final screening: prompt input with full-text variable and JSON output.

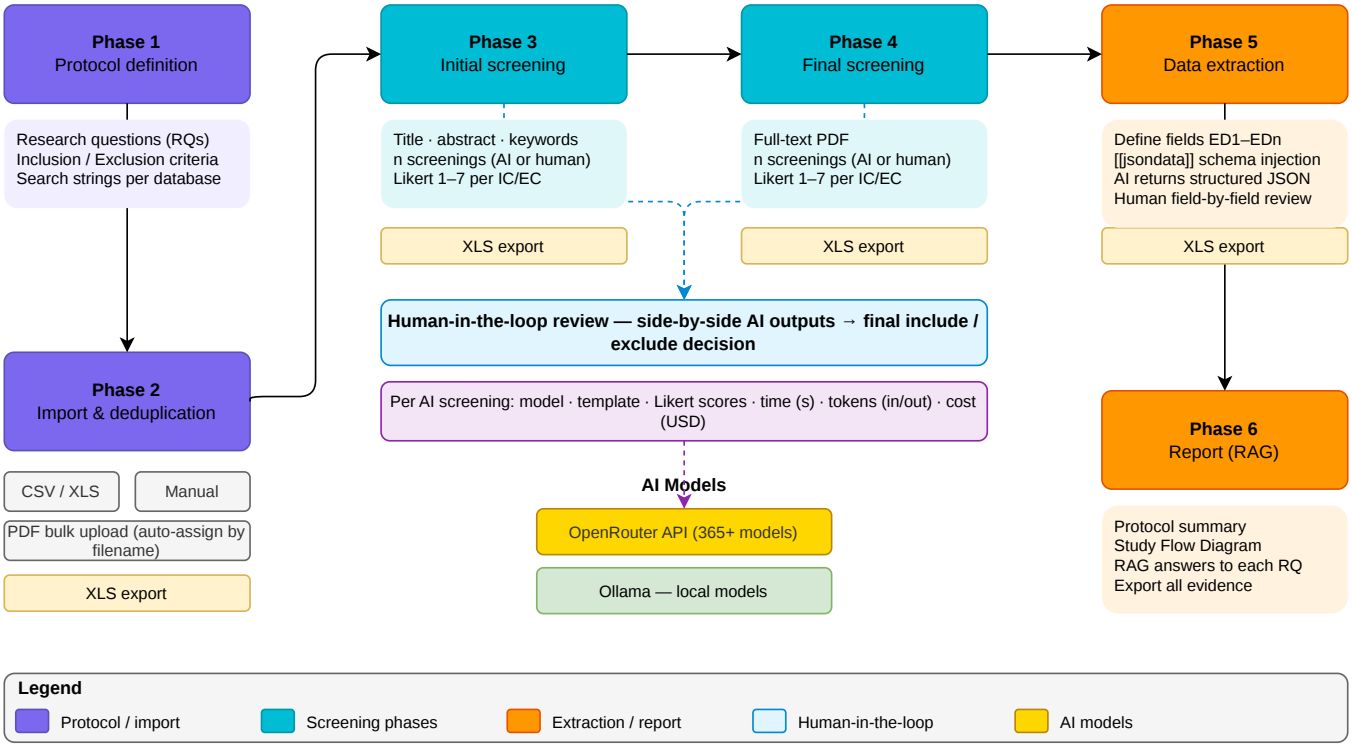

Figure 2: AI LitReview six-phase SLR workflow, from protocol definition to RAG-based report generation.

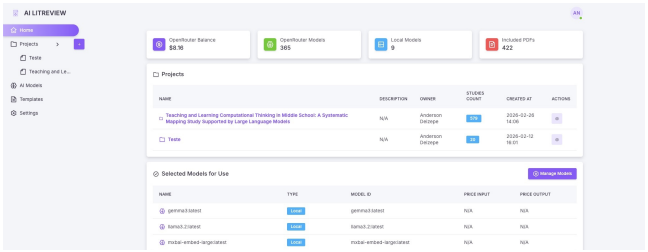

Figure 3: AI LitReview home dashboard: OpenRouter balance, model counts, included PDFs, project list, and selected models with pricing.

Listing 3: Data extraction: schema injection via `[[jsondata]]` and structured output.

```
-- INPUT --
Extract the following fields from the study. [...]
[[jsondata]]
Study full text: [[fullstudy]]
-- OUTPUT --
{
  "ED1":  "S42",
  "ED10": "Journal article",
  "ED11": "Empirical",
}
```

```
-- INPUT --
Assume you are a software engineering researcher. [...]
[[fullstudy]]
[[inclusioncriteria]]
[[exclusioncriteria]]
Return ONLY valid JSON.
-- OUTPUT --
{
  "criteria": [
    { "id": "IC1", "score": 7,
      "justification": "Empirical study with..." },
  ],
  "overall": "include"
}
```

4.4 Study Selection: Initial and Final Selection

As previously described, the study selection follows two stages (initial and final selection). Both stages support unlimited independent *selections*, each configured with a model type (Human, AI OpenRouter, AI Local), a model, a template, and a responsible researcher. Selections are processed asynchronously with the progress tracked per study (Figure 4).

To make a final selection decision, researchers review all LLM-generated selection output side by side, including Likert scores for each selection criterion, before marking each study as included or excluded. In both stages, the boundary between automation and support is explicit, the AI automates the reading effort, scoring each criterion and justifying the score, but never includes or excludes a study on its own; the decision is always made by a human researcher and logged as such. For each LLM-supported selection activity, the

tool also records metadata such as processing time (s), input tokens, output tokens, and execution cost (USD).

Regarding reliability, each screening (human or AI) is treated as an independent rater over the same set of studies. The per-criterion scores and decisions of all raters can be exported to XLS, enabling inter-rater agreement (e.g., Cohen's kappa) to be computed between AI runs, between AI and human raters, or among humans, with the AI explicitly counted as one of the raters. Built-in agreement metrics are planned as future work.

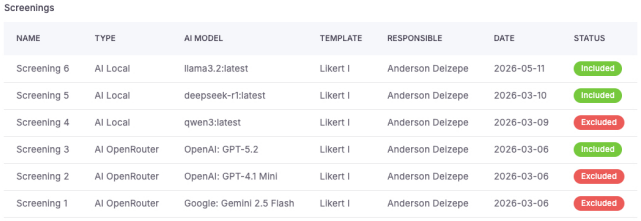

Figure 4: Multiple AI screenings, showing different models, progress, and include/exclude results.

4.5 Data Extraction

For data extraction, extraction fields (e.g., `ED1-ED31`) are defined as part of the protocol, each field has an identifier, a name, an expected value or type, and a guidance note describing what to look for in the study. The extraction fields are inserted in the prompt by the `[[jsondata]]` variable; structured JSON is returned by the LLM (Listing 3). Beyond schema injection, the researcher is supported in three ways, multiple extractions with different models can be run over the same studies and compared field by field; each AI-suggested value is displayed alongside the study for verification; and each value is accepted, edited, or overridden individually, no extracted value is persisted without human confirmation.

4.6 Report Generation

Finally, the tool generates a report containing the SLR protocol, a Study Flow Diagram (Sankey chart), and answers to the RQs produced through an RAG pipeline based on the full texts of the included studies. The synthesized answers are presented as AI-generated drafts, and researchers are expected to verify them against the source studies before use. All evidence can be exported.

5 Usage Example

To illustrate AI LitReview, we describe its application in a Systematic Mapping Study (SMS) on *Teaching and Learning Computational Thinking in Middle School*. The project comprises 579 studies imported from multiple databases. After removal of duplicate and exclusion criteria, 577 studies remained. Six screenings were performed in the initial selection activity using different models: Gemini 2.5 Flash, GPT-4.1 Mini, GPT-5.2, and three local Llama 3.2 runs. The final selection used GPT-5.2 and Gemini 2.5 Flash on full PDFs. Across all screenings, the tool recorded token usage and cost per study. For example, the GPT-5.2 final selection consumed 30,400 tokens per article at $0.06 each. Of 469 studies that passed the initial selection, 282 were included after final selection and all proceeded to extraction. The extraction activity defined 31 fields (ED1-ED31) that cover document type, study nature, educational context, approach, benefits, limitations, tools, assessment methods, challenges and recommendations. The RAG-based report synthesized answers to five (5) RQs from the included PDFs. The complete dataset, selections, and report are visible on the tool's home dashboard, which also shows the OpenRouter balance ($8.16 remaining), 365 available API models, and nine (9) active local models.

6 Conclusion and Future Work

This paper presented AI LitReview, a web-based tool that integrates LLM support throughout the SLR process. Its key contributions are: (i) multi-phase LLM assistance with configurable prompt templates; (ii) support for local models via Ollama and API models via OpenRouter; (iii) human decision supported by multiple LLM-generated selections, with the AI acting as an advisory rater and never as the decision-maker; (iv) study cost, token, and latency tracking for reproducibility and cost-awareness; (v) RAG-based report generation based on the included study content; and (vi) complete collaboration, auditability, and XLS export in each phase.

Several open challenges identified in the literature on LLM-assisted SLRs [6] directly inform our future work agenda. Prompt optimization remains an important research direction, particularly in investigating how variations in wording, few-shot examples, and context framing influence selection accuracy across different SE domains. Replication support will be strengthened by storing exact model versions, parameter settings, and run metadata alongside each screening, enabling researchers to reproduce or audit any LLM-supported decision.

It is also planned (i) to embed evidence-based guidance on which models and prompting strategies best suit each SLR activity, informed by benchmarks such as SESR-Eval [11] and by the tool's own accumulated execution logs; (ii) to provide built-in inter-rater reliability metrics (e.g., Cohen's kappa) that treat each AI screening as a rater; (iii) to add support for snowballing (backward and forward citation tracking); (iv) to expand the RAG pipeline with more advanced chunking and retrieval strategies; and (v) to add support for systematic mapping visualizations (e.g. bubble graphs).

An empirical evaluation with SE researchers will assess the tool's impact on SLR efficiency, accuracy, and perceived usability.

Artifact Availability

AI LitReview is publicly available at https://ailitreview.dzp.net.br. The source code, the demo video, and the additional materials are available at https://doi.org/10.5281/zenodo.20387266 under the MIT license. The source code is licensed under the MIT license and is available on GitHub at https://github.com/Deizepe/AI_LitReview/.

Acknowledgements

The authors thank the researchers who used AI LitReview in their SLRs and provided valuable feedback throughout its development.

Professor Katia Romero Felizardo is funded by a research grant from the Brazilian National Council for Scientific and Technological Development (CNPq), Grant #302339/2022-1.

References

[1] David Bowes, Tracy Hall, and Sarah Beecham. 2012. SLuRp: A Tool to Help Large Complex Systematic Literature Reviews Deliver Valid and Rigorous Results. In *Proceedings of the 2nd International Workshop on Evidential Assessment of Software Technologies (EAST)*. ACM, 33–36. doi:10.1145/2372233.2372243

[2] O. Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. *Journal of Systems and Software* 80, 4 (2007), 571–583. doi:10.1016/j.jss.2006.07.009

[3] Tom B. Brown et al. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.

[4] D. Comis et al. 2025. Thoth 2.0: Advancing an RSL Tool for Enhanced Snowballing Support. In *Anais do XXXIX Simpósio Brasileiro de Engenharia de Software (SBES)*. Sociedade Brasileira de Computação (SBC).

[5] Sandra Fabbri, Cleiton Silva, Érico Hernandes, Francisco Octaviano, Andreia Di Thommazo, and Adauto Belgamo. 2016. Improvements in the StArt tool to better support the Systematic Literature Review Process. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE)*. doi:10.1145/2915970.2916013

[6] Katia Romero Felizardo, Anderson Deizepe, Daniel Coutinho, Genildo Gomes, Maria Meireles, Marco A. Gerosa, and Igor Steinmacher. 2025. On the Difficulties of Conducting and Replicating Systematic Literature Reviews Studies Using LLMs in Software Engineering. In *Proceedings of the 2025 IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering (WSESE)*. IEEE, 20–23. doi:10.1109/WSESE66602.2025.00010

[7] Katia Romero Felizardo, Márcia Sampaio Lima, Anderson Deizepe, Tayana Uchôa Conte, and Igor Steinmacher. 2024. ChatGPT Application in Systematic Literature Reviews in Software Engineering: An Evaluation of Its Accuracy to Support the Selection Activity. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '24)*. ACM, 25–36. doi:10.1145/3674805.3686666

[8] Katia Romero Felizardo, Elisa Yumi Nakagawa, Sandra Camargo Pinto Ferraz Fabbri, and Fabiano Cutigi Ferrari. 2017. *Revisão Sistemática da Literatura em Engenharia de Software: Teoria e Prática.* Elsevier.

[9] Ana M. Fernández-Sáez, Marcela Genero, and Francisco P. Romero. 2010. SLR-TOOL: A Tool for Performing Systematic Literature Reviews. In *Proceedings of the 5th International Conference on Software and Data Technologies (ICSOFT)*. SciTePress, 157–166. doi:10.5220/0003003601570166

[10] Vitor Freitas. 2013. Parsifal: Perform Systematic Literature Reviews. https://parsif.al.

[11] Aleksi Huotala, Miikka Kuutila, and Mika Mäntylä. 2025. SESR-Eval: Dataset for Evaluating LLMs in the Title-Abstract Screening of Systematic Reviews. In *Proceedings of the 19th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 218–229. doi:10.1109/ESEM64174.2025.00053

[12] Aleksi Huotala, Miikka Kuutila, Olli-Pekka Turtio, and Mika Mäntylä. 2025. AISysRev – LLM-based Tool for Title-abstract Screening. arXiv:2510.06708. doi:10.48550/arXiv.2510.06708

[13] Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering.* Technical Report EBSE-2007-01. Keele University and Durham University.

[14] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. 2015. *Evidence-Based Software Engineering and Systematic Reviews.* Chapman and Hall/CRC. doi:10.1201/b19467

[15] Patrick Lewis et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, Vol. 33. 9459–9474.

[16] Luciano Marchezan, Guilherme Bolfe, Elder Rodrigues, Maicon Bernardino, and Fábio Paulo Basso. 2019. Thoth: A Web-based Tool to Support Systematic Reviews. In *Proceedings of the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–6. doi:10.1109/ESEM.2019.8870160

[17] Christopher Marshall and Pearl Brereton. 2013. Tools to Support Systematic Literature Reviews in Software Engineering: A Mapping Study. In *Proceedings of the 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 296–299. doi:10.1109/ESEM.2013.32

[18] Mourad Ouzzani, Hossam Hammady, Zbigniew Fedorowicz, and Ahmed Elmagarmid. 2016. Rayyan—A Web and Mobile App for Systematic Reviews. *Systematic Reviews* 5, 1 (2016), 210. doi:10.1186/s13643-016-0384-4