

AI-ENABLED SOFTWARE REQUIREMENTS SPECIFICATION FOR BANKING CHATBOT SYSTEMS

Ansu Mary JACOB , Jolanta MILIAUSKAITĖ 

*Department of Information Systems, Faculty of Fundamental Sciences, Vilnius Gediminas Technical University,
10223 Vilnius, Lithuania* 

Article History:

- received 30 April 2026
- accepted 10 June 2026

Abstract. Software Requirements Specification plays an important role in the success of software development projects by providing clear, structured, and consistent documentation of requirements. However, traditional SRS processes are manual, time-consuming, ambiguous, and inconsistent. This paper proposes an AI-enabled software requirements specification framework using Retrieval-Augmented Generation. The method combines Large Language Models with domain-specific knowledge retrieval to generate structured, coherent, and contextual requirement specifications. The proposed approach integrates document management, semantic retrieval, and generative reasoning to improve requirements quality, reduce ambiguity and enhance traceability. Validation was conducted using a banking chatbot case study. Experimental evaluation showed that the proposed approach achieved 0.75 precision, 1.00 recall, 0.857 F1-score and 0.75 accuracy for Functional Requirement identification on a manually labelled 12-sample evaluation subset. However, the system did not correctly identify Non-Functional Requirements in the current evaluation, indicating the need for a refined FR/NFR classification strategy in future work. The results demonstrate that Retrieval-Augmented Generation provides a practical and scalable solution for modern intelligent requirements engineering systems.

Keywords: software requirements specification, artificial intelligence, retrieval-augmented generation, large language models, natural language processing, functional requirements, non-functional requirements.

✉Corresponding author. E-mail: ansu-mary.jacob@stud.vilniustech.lt

1. Introduction

A key stage of software development is ensuring system functionality, quality, and compliance with client expectations. A well-defined specification provides a structured framework for design, implementation, and validation. However, traditional SRS processes are manual, time-consuming, and prone to inconsistency, incompleteness, and ambiguity. Given the dynamic, data-driven, and constantly changing requirements of modern AI-based systems, addressing these challenges has become increasingly important.

Recent research (Maalej et al., 2023; Zowghi et al., 2023), indicates that traditional specification approaches cannot manage the complexity of AI systems, especially regarding non-deterministic behaviour, evolving datasets and stakeholder-driven changes. Without structured automation, specification procedures frequently lead to uncertain requirements, higher development costs, and reduced system reliability.

Furthermore, unstructured natural language inputs from customers introduce uncertainty, making it challenging to produce requirement specifications that are reliable and consistent (Liu et al., 2022). Artificial Intelligence (AI) techniques, particularly Natural Language

Processing (NLP) and Large Language Models (LLMs), are increasingly used to automate requirements extraction and specification to address these challenges (Ahmad et al., 2023; Habiba et al., 2024). Recent advancements in Retrieval-Augmented Generation (RAG) allow the combination of generative models with external knowledge sources, resulting in more accurate and context-aware semantics (Siddeshwar et al., 2024; Taj et al., 2025).

However, standalone LLMs often lack domain-specific grounding and are prone to hallucinations, making them less reliable for requirements formulation. Therefore, the research problem addressed in this paper is “How can Software Requirements Specification processes be improved using AI-based development?”.

In this paper, we propose an AI-enabled SRS approach based on RAG, which combines domain-specific Knowledge Bases (KBs) with LLMs to generate structured and context-aware requirement specifications.

The aim of this paper is to improve the SRS processes by applying AI-based techniques for automated and adaptive specification generation.

The main contribution and novelty of this research are as follows:

1. A structured AI-enabled SRS framework using RAG and LLMs for automated specification generation.
2. A continuous feedback-driven learning process used to enhance requirement quality over time.
3. Integration of domain-specific KBs to increase contextual accuracy and lessen ambiguity.
4. Experimental validation using a banking chatbot case study demonstrating improved correctness, completeness and relevance of specifications.
5. Evaluation of requirements quality improvement using AI assistance.

The rest of the paper is organized as follows. Related work is discussed in Section 2. Section 3 introduces the proposed AI-enabled RAG based SRS approach. Section 4 presents the case study and implementation results. Section 5 provides the discussion of the findings, limitations, and interpretation of the experimental results. Finally, in Section 6 conclusions and future work are presented.

2. Related works

The growing use of AI in software systems creates serious issues due to traditional RE methods. While AI-based systems are inherently data-driven, dynamic and non-deterministic, conventional SRS approaches are primarily designed for deterministic and stable systems. As a result, current research has focused on improving SRS practices using AI approaches, including NLP, ML, and LLMs (Habiba et al., 2024; Liu et al., 2022).

Various studies have explored the use of ML and NLP approaches for automating requirement extraction and classification, with recent work highlighting the importance of AI-based techniques for distinguishing between requirement types such as FR and NFR (Kaur & Kaur, 2024; Liu et al., 2022). Feature engineering techniques like TF-IDF and Bag-of-words are often combined with traditional models such as Support Vector Machines (SVM), Naïve Bayes (NB) and Random Forest (RF) (Kaur & Kaur, 2024; Liu et al., 2022; Sofian et al., 2022). While these methods work well in controlled environments, they are unable to capture the deep semantic

relationships in natural language, which limits their ability to handle uncertainty and contextual variations in requirements. Deep Learning (DL) approaches, especially transformer-based models like BERT, have been developed to overcome these limitations.

Due to their capability to understand contextual semantics, BERT-based models significantly outperform traditional ML methods in requirements classification tasks, as shown in studies such as (Siddeshwar et al., 2024; Taj et al., 2025). Additionally, explainability techniques like LIME improve the interpretability of these models. Despite these advancements, these approaches still rely on static datasets and cannot adequately support ongoing learning or adapt to the evolving needs that are a key characteristic of AI-based systems.

Transfer learning methods outperform the traditional ML and DL models in terms of accuracy, in accordance with systematic research evaluations (Kaur & Kaur, 2024). However, these strategies often lack reliability in complex real-world situations and require substantial computational resources. Similarly, collaborative learning approaches (Wang et al., 2024) promote transparency and enable shared learning, but they also have drawbacks, such as lower precision and increased system complexity.

A wide variety of approaches have already been proposed to enhance RE practices for AI systems, beyond algorithmic techniques. The GR4ML concept (Nalchigar et al., 2021), for example, presents a goal-based planning approach that bridges the gap between technological implementation and organizational goals. While this system promotes traceability and consistency, its scalability is limited due high resource demands, the need for specialised knowledge, and a lack of automation.

Recent research has also emphasized the importance of ethical and responsible AI in requirements engineering. According to Maalej et al. (2023), RE must incorporate principles of fairness, accountability, and transparency. While these approaches represent significant innovations, they require support systems and practical application procedures. Similarly, studies such as Zowghi et al. (2023) highlight the need for new methods in software engineering to address AI-specific challenges, such as system-wide autonomy and continuous learning, but these studies do not offer concrete solutions.

Recent research studies assess how LLMs may affect RE approaches, particularly in requirement elicitation, documentation, analysis and specification support (Borg & Borg, 2024; Cheng et al., 2026). The authors of the paper (Ferrari, 2023) discusses how LLMs render prompt-driven requirements synthesis and implementation viable. Even though LLMs improve efficiency and ease of access, they can result in significant issues, such as lack domain grounding, unreliable performance, and hallucinations. These drawbacks render them inconsistent in domain-dependent systems and highly sensitive to safety.

RAG is a potential hybrid synthesis that integrates conventional generative models and data extraction to address these types of problems. RAG enhances context-related clarity and reduces hallucinations by acquiring relevant, domain-specific information in advance of generating responses. These results make it particularly suitable for assignments comprising specification drafting, where integrity and subject proficiency are required. Yet the scope of modern RAG-based approaches remains narrow, and they usually lack standard mechanisms for continuous improvement and perpetual learning (Habiba et al., 2024).

Contextual embedding and the ability to adapt are two essential elements by which requirement specification approaches may be examined, as gathered from previous research. The level of significance with which an approach integrates specific domain data is expressed through the corresponding contextual embedding aspect, which ranges from RAG-based systems with continuous knowledge integration to typical ML approaches with limited lexical awareness.

As summarised from the previous analysis, contextual grounding and versatility are two key elements by which requirement specification approaches may be evaluated. The way a system blends domain-specific data is indicated by the contextual grounding dimension, which ranges from RAG-based systems with dynamic knowledge fusion to traditional ML approaches with limited semantic understanding. The system's ability to evolve over time from rigid models with pre-determined training data to systems with active learning and feedback-driven enhancements is reflected in the flexible feature.

Therefore, this research addresses the identified limitations by proposing an AI-enabled SRS approach based on RAG. The suggested approach combines a feedback-driven continuous learning mechanism, LLM-based generation, and domain-specific KB retrieval. This approach reduces ambiguity in client inputs, improves the correctness and clarity of requirements descriptions, and facilitates incremental modifications over time by blending semantic retrieval with context-specific synthesis. The method's performance as a scalable and adaptable solution for automated requirement planning in dynamic environments has been demonstrated through a banking chatbot case study.

3. Proposed approach

By uniting domain-specific data retrieval, LLM-based synthesis, and a feedback-oriented persistent learning approach, the proposed chatbot system aims to automate and strengthen demand-based specification.

In the standard system illustrated in Figure 1, users can interact in real time by sending requests to the chatbot. An AI model processes these requests and executes the RAG approach to search for appropriate data from an organised KB. The user is then provided with a response that is contextually precise and comprehensive based on the retrieved subject matter. After receiving the response, the user assesses its value; if the response is satisfactory, the interaction ends. If not, the system logs a thorough analysis and timeline. The system trainer collects and examines this feedback to identify knowledge gaps.

New requirements are established and sorted into Functional Requirements (FR) and Non-Functional Requirements (NFR) because of this research. A confusion matrix and traditional metrics of precision, recall, F1-score and accuracy, as outlined in Eqs. (1)–(4), are applied in this research to evaluate the level of excellence classification.

Chatbots are trained or refined to align with these higher quality insights, which are then used to revise and improve the KB. Over time, the chatbot becomes more reliable and contextually responsive with each response cycle.

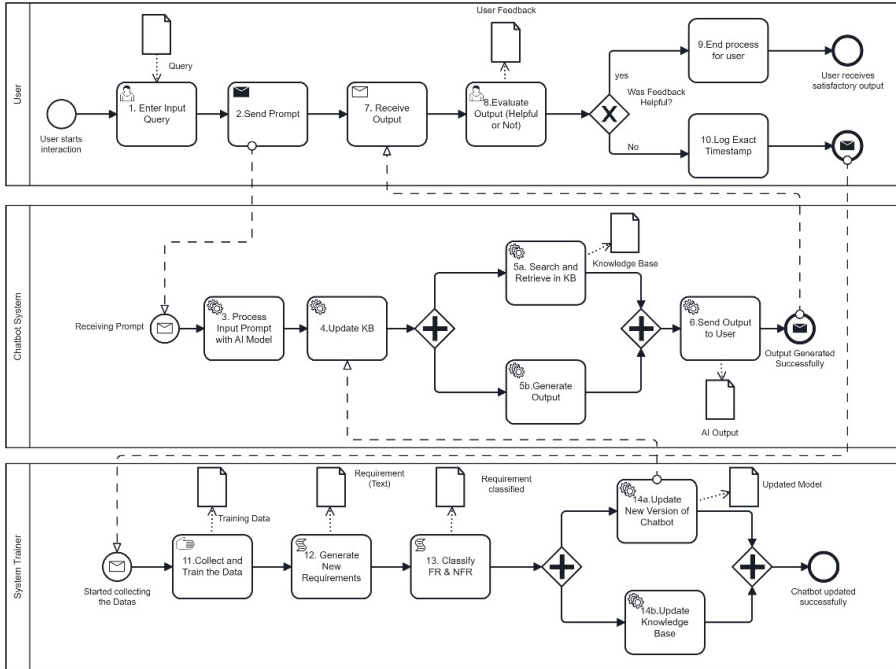


Figure 1. General schema of proposed approach

The general schema of the proposed approach to the above process in the AI-enabled SRS approach is divided into three interacting layers i.e., system trainer, chatbot system, and user.

The procedure begins at the Chatbot AI System layer when the user issues a prompt. The “process input prompt using AI model” function processes input by interpreting the prompt and preparing a reaction. The “update KB” operation is then performed, updating the KB with the most recent data and making it ready for retrieval. After that, the procedure is divided into two independent concurrent tasks. In order to retrieve essential domain-specific information from the KB, the system first performs “KB search and retrieval”. Secondly, the system “generates output” in which the AI model uses contextual data obtained from the processed prompt to provide a response. The system then performs “send output to user” when these two branches have synchronized. Following this, the process reaches the “output generated successfully” event, which signifies that the user has received the response.

The System Trainer layer, which is responsible for handling ongoing learning and system change, receives feedback and logged interactions when the user indicates that the output is not useful. “Start to collect data” is the first step in the training cycle, during which unsuccessful or deficient interactions are gathered. System prepares training data by combining interaction history, prompt-response pairs, and feedback data in the subsequent phase “collect and train data”. The system uses this data to execute “new requirements generation” which identifies missing, uncertain, or incomplete requirements based on patterns of customer dissatisfaction.

Classifying the FR and NFR activity receives the freshly written requirement text and breaks it into FR and NFR. The processes essentially fork into two parallel updating tasks once again, following classification. In order to “update the new version of the chatbot”, a branch modifies the chatbot’s logic or model in accordance with newly discovered needs. The “update knowledge base” is performed by another branch, which fills detected knowledge gaps by expanding or correcting the domain KB.

The approach reaches the “chatbot updated successfully” event once both updates have completed, indicating that the system has been enhanced and is prepared for further interactions. The proposed approach supports the end-to-end SRS process by incorporating a variety of standard AI and NLP components.

3.1. Retrieval-augmented generation mechanism

The RAG approach, which blends neural text analysis with semantic retrieval (Fan et al., 2024; Lewis et al., 2020). The RAG model pulls relevant information from a distant KB before producing a response, in contrast with independent LLM-based systems that only use pre-loaded knowledge. This basing process ensures that created requirements are based on recent, domain-relevant knowledge and minimises hallucinations.

3.2. Text normalization and preprocessing

The input data need to be cleaned and standardised before knowledge can be established or accessed. File distribution, data cleansing, content refinement and basic classification are all handled by the initial step of the suggested structure. Heterogeneous inputs such as interview notes, policy documents and user-generated text are refined at this step.

To increase the reliability of subsequent retrieval and standard extraction, the aim is to reduce the noise and enhance organisational stability. When needed, this procedure also helps with the initial classification of information linked to requirements, enabling the method to separate potentially meaningful requirement components from superfluous linguistic material. Before the data is added to the knowledge pipeline, the procedure enables manual inspection if the input quality is unsatisfactory.

3.3. Knowledge base construction and enrichment

The RAG system’s primary source of verified domain information is the KB. It keeps organised data from papers, previous specifications, domain rules and changes produced by user feedback. The KB’s quality is vital, as the quality of the requirements is significantly affected by the accuracy and completeness of the retrieved information.

3.4. LLM-based requirement extraction and structuring

The system uses an LLM to convert the received data and the user query to a formatted requirements specification when essential context has already been gathered. This is the system’s primary specification task. The LLM is prompted to determine the customer’s intent, uncover the root desire, and restate it as a well-defined and unambiguous requirement specification that can be codified.

In order to guarantee that outcomes follow basic requirements specification conventions like organised, measurable and understandable terms (e.g., “The system must...”), the structure places a high value on standard format normalisation and pattern compliance. Upon generation, the output is scrutinized for relevance, consistency and completeness. The system allows for manual assessment, cross-verification, and re-prompting if the result obtained is insufficient.

The process provided in Figure 1 is assigned to a sequence of analyzing stages associated with deliverables in order to offer clear details of the way that the proposed system performs. The outline links each phase to the approach and analysis process, and Table 1 lists the main duties, transitional findings, and goals for each phase.

Table 1. Stepwise mapping of the proposed AI-enabled SRS approach linking steps

Diagram Stages	Process Step	Process activity	Input	Output / Result
1–2	Step 1: Input Query and Retrieved Knowledge	User enters query and sends prompt	User query	Submitted query
3	Step 2: prompt construction.	Process input prompt with AI model	Submitted query	Processed query
4	Step 1: Input Query and Retrieved Knowledge	Update/check knowledge base	Processed query	Prepared KB state
5a	Step 1: Input Query and Retrieved Knowledge	Search and retrieve in knowledge base	Processed query	Retrieved KB context
5b	Step 3: Generated Requirement Output	Generate output	Prompt + KB context	AI-generated output
6–8	Step 3: Generated Requirement Output	Send, receive, and evaluate output	AI-generated output	Chatbot response and feedback status
9–10	Step 5: Knowledge Gap and Feedback Update	End process or log timestamp	Feedback status	Success or feedback record
11	Step 5: Knowledge Gap and Feedback Update	Collect and train the data	Feedback record	Training data
12	Step 3: Generated Requirement Output	Generate new requirements	Training data	Requirement (Text)
13	Step 4: FR/NFR Classification and Evaluation	Classify FR & NFR	Requirement (Text)	FR/NFR labels
14a	Step 5: Knowledge Gap and Feedback Update	Update new version of chatbot	Classified requirements	Updated Model
14b	Step 5: Knowledge Gap and Feedback Update	Update knowledge base	Classified requirements	Updated KB

The above mapping clearly shows how user interactions are transformed into structured requirements, evaluated, and used to continuously improve both the KB and the chatbot model through the SRS process.

3.5. Proposed system architecture

For the chatbots, the systems being tested are the baseline chatbot and the enriched chatbot. A typical RAG system serves as a foundation chatbot. Although its learning module is deactivated, it has been set up using the same base LLM and starting KB as the treatment group.

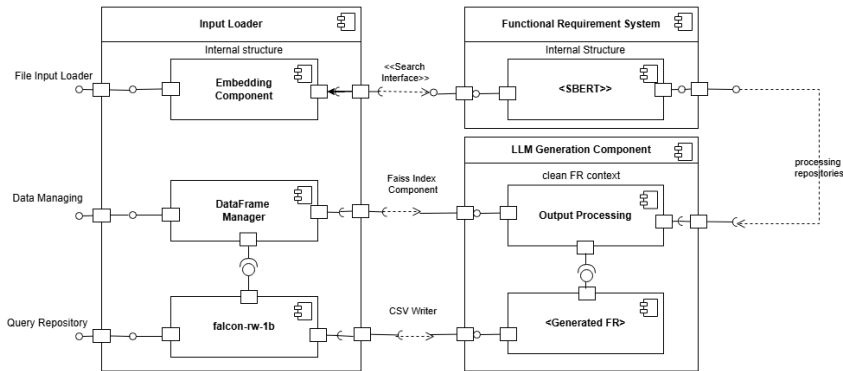


Figure 2. Architecture of the proposed AI-Enabled functional requirement generation system

For the system trainer module and the chatbot interface of the enhanced RAG chatbot. The LLM, KB and automated requirements pipeline are all fully functional parts of this system. The FR generating system architecture of the proposed AI-enabled system is shown in Figure 2 and is discussed further. This is an AI-enabled technology that assists in generating the FR from customers.

Using the input loader, the system first receives the input data. Data concerning a requirement is often incorporated in this entry. This text is transformed into numerical illustrations via the embedding component. In order for the system to fully understand what the requirements constitute. Conversely, the DataFrame Manager (System Trainer) arranges the data systematically to enable its handling and storage. Metadata from all of these processes is kept in a query repository, which serves as a hub for the system's required data.

By comparing the meanings rather than precise words, the FR generating system employs SBERT to determine the most relevant requirements facts. Beyond that, this data is passed to the LLM generation feature, which uses natural language to provide clear and essential operational requirements. These generated requirements are then cleaned, modified, and saved into a file by the output processing module. So that we may evaluate or reuse it with convenience.

Ultimately, by extracting the software FR using the enhanced RAG chatbot, this system reveals how the AI-enabled SRS with a RAG may reduce manual effort and increase accuracy.

4. Case study and implementation results

The case study and the research results from the review are discussed in this section. In authentic AI-enabled system contexts, the goal of this case study is to verify whether the proposed approach can successfully enable continuous requirements extraction, discover

knowledge gaps from user interactions, and incrementally enhance the requirement quality by improving the SRS processes.

The research analyses Banking Chatbot data in the banking domain to determine how efficiently SRS procedures are implemented in agile, knowledge-driven AI systems (Adamopoulou & Moussiades, 2020). For instance, the 145 QA combinations were used in the case study, and some dataset samples are presented in Figure 3 below. The banking chatbot KB used in this study comprises 145 query-response pairs in JSON format. These 145 QA pairs were used as the domain-specific KB for retrieval and requirement generation.

However, the quantitative FR/NFR classification evaluation was conducted on a smaller manually labelled subset of 12 representative samples. This subset was selected to demonstrate the evaluation procedure and to provide an initial assessment of the prototype. Therefore, the 145 QA combinations refer to the retrieval KB, while the 12 samples refer only to the labelled evaluation set used for the confusion matrix and performance metrics. This limitation has been acknowledged, and future work will extend the evaluation to the full dataset and additional domains. The financial services sector is an ideal test ground for demonstrating the performance of AI-enabled SRS process generation, since the industry involves high levels of complexity, criticality, strict guidelines and imprecise user questions. An essential part of our RAG conception is a domain-specific banking KB, largely controlled via JSON files.

The system can successfully obtain the information because each JSON item holds a structured query-response pair. This format supports logical matching and improves the correctness and context-based nature of the ensuing requirements statements.

```
[
  {
    "Query": "How can I open a new bank account?",
    "Response": "To open a new bank account, visit our website and click on 'Open Account' or visit any of our branches with your identification documents."
  },
  {
    "Query": "What do I need to provide to open a bank account?",
    "Response": "You will need to provide a valid ID, proof of address, and your Social Security Number or equivalent identification number."
  },
  {
    "Query": "How can I check my account balance?",
    "Response": "You can check your account balance by logging into our online banking portal or mobile app. You can also check your balance at an ATM."
  },
  {
    "Query": "How do I transfer money between accounts?",
    "Response": "Log into your online banking account or mobile app, select 'Transfer Funds,' choose the accounts, and enter the amount you wish to transfer."
  },
  {
    "Query": "Can I set up direct deposit for my paycheck?",
    "Response": "Yes, you can set up direct deposit by providing your employer with your account number and our bank's routing number."
  },
  {
    "Query": "How do I report a lost or stolen debit card?",
    "Response": "Report a lost or stolen debit card immediately by calling our customer service at 1-800-123-4567 or through our mobile app."
  },
  {
    "Query": "What should I do if I forget my online banking password?",
    "Response": "Click on 'Forgot Password' on the login page and follow the instructions to reset your password using your registered email address."
  }
]
```

Figure 3. Banking chatbot dataset sample

The experiment follows a main step-by-step pipeline: input query selection, knowledge base retrieval, prompt construction, LLM-based requirement generation, FR/NFR classification, and evaluation as shown in Table 1.

Step 1. Input query and retrieved knowledge

The user query, which correlates to the JSON-based banking KB, is the starting point in the proposed structure. The system extracts the most relevant stored information using the

RAG retrieval process to provide the ensuing demand generation step with a situational basis. Sample QA pairs from the banking KB are shown in Table 2.

Table 2. Example of query retrieval from the banking knowledge base

No.	Input Query	Retrieved KB response
1	How can I open a new bank account?	To open a new bank account, visit our website and click "Open Account" or visit a branch with identification documents.
2	How can I check my account balance?	You can check your account balance by logging into online banking, mobile app, or ATM.
3	Can I set up direct deposit for my paycheck	Yes, direct deposit can be set up by providing the employer with account and routing details.

Every input to a query is appropriately correlated with an applicable KB response, as shown by the outputs in Table 2. The LLM employs this extracted context as its primary input, which permits the system to provide more accurate and domain-neutral requirements specifications in a subsequent phase. The retrieval process follows the RAG paradigm which combines retrieval with generative model (Lewis et al., 2020).

Step 2. Prompt construction

Prompt creation comes next after acquiring the relevant KB entry. From this stage, the system combines the user query, the obtained related response, and a predefined proposal pattern to help the LLM build an orderly needs specification. An example of how the prompt is brought along to generate the requirement is shown in Table 3.

Table 3. Prompt construction used for requirement generation

Component	Content
User query	How can I open a new bank account?
Retrieved context	To open a new bank account, visit the website and click "Open Account" or visit a branch with identification documents.
Prompt Construction	Generate a clear software requirement using the format "The system shall..."
Expected output type	Functional Requirement

The prompt-based format guides the LLM to generate a formal requirement rather than an interactive response, as Table 3 depicts. The LLM was treated as a black-box generator in this research, which means that none of the parameter variations or corrections were applied. In contrast, search-focused prompts and the obligatory "The system shall..." requirement format pushed the quality of output.

Step 3. Generated requirement output

The organized cue is handed over to the LLM to generate a requirement specification after it has been built. Throughout this stage, the blueprint is transformed into a formal software requirement using the "The system shall..." format, which specifies for the user's intent and the collected knowledge of banking. Examples of preliminary requirement response generated by the system are shown in Table 4.

Table 4. Intermediate requirement generation results

User query	Generated Requirement	Requirement type
How can I open a new bank account?	The system shall have the following features in order to enable me to open a new bank account.	FR
How can I check my account balance?	The system shall support customers to view their account balance using the following method:	FR
Can I set up direct deposit for my paycheck?	The system shall allow the user to set up direct deposit for their paycheck.	FR

The resulting outputs are transformed from natural language queries into structured requirements specifications appropriate for the RE process as shown in Table 4. Based on the extracted domain-specific information, these initial findings show that the proposed method can reliably generate FR specifications, which are then sent to the classification and evaluation phase.

Step 4. FR/NFR classification and evaluation

After the requirements generation, the outputs are converted from natural language queries into structured requirements specifications appropriate for the RE process, as shown in Table 5.

Based on the gathered domain knowledge, these early findings show that the proposed approach can reliably generate FR specifications, which are then sent to the standard classification and evaluation process.

Table 5. Sample classification result

Generated requirement	True label	Predicted label	Result
The system shall allow users to open a new bank account through the website or branch.	FR	FR	Correct
The system shall provide secure password recovery for online banking users.	FR	FR	Correct
The system shall provide information about savings account interest rates.	NFR	FR	Incorrect

As shown in Table 5, the proposed system identifies several FRs, but some NFRs are misclassified as FR. The classification performance was assessed using precision, recall, F1-score, and accuracy based on the confusion matrix presented later in this paper. These findings show that the system does a good job of identifying FR, but further work is required to identify statements linked to NFR.

Step 5. Knowledge gap and feedback update

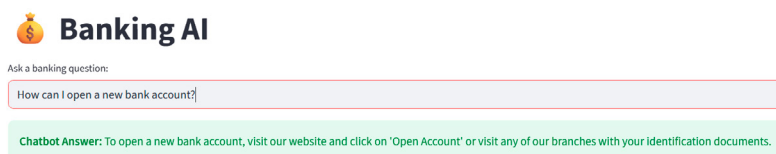
The last stage of the approach is all about feedback-based ongoing development. The omitted data is not considered if the system is unable to obtain an accurate response or only provides a partial response. However, these instances are addressed as knowledge gaps, which are noted and applied to enhance the knowledge base and create future requirements. Examples of this feedback-based adaptation presented in Table 6.

Table 6. Example of knowledge gap handling

User query	System Status	User feedback / missing information	Action taken
What is the policy for foreign transactions?	No exact answer found	User requested foreign transaction fee information	New knowledge gap logged
What is the process to dispute a transaction?	Partial answer found	Missing dispute procedure details	Requirement updated and KB refined

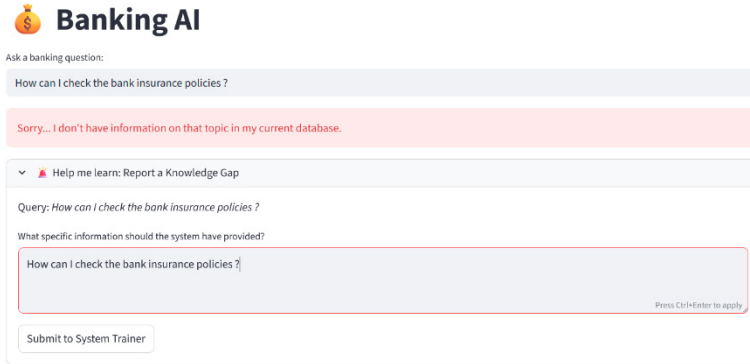
As shown in Table 6, the incomplete or unsuccessful responses are transformed into useful learning signals for the system. With the assistance of this method, the KB may be progressively improved, making the chatbot more precise, context-aware and more suited to changing user queries over time. The proposed enhanced chatbot differs notably in this point.

The JSON-formatted dataset serves as a grounding source for the LLM to satisfy thematically appropriate requirements by compatible user “Query” with corresponding “Responses”. The Streamlit interface is used to create a web-based prototype of the proposed AI-enabled SRS system. This version showcases how the system combines with continuous requirements gathering and modification with knowledge-based question answering. The KB upload panel enables the person in charge to constantly enhance the system’s domain knowledge. Retrieval relevance, requirement accuracy, FR/NFR classification, and feedback-based modification are among the parameters evaluated. The interface supports submitting structured JSON files with QA pairings related to banking. The system feeds the data into the knowledge retrieval pipeline after verifying the file format. The banking AI interaction interface is shown in Figure 4 below.

**Figure 4.** AI chat interface

For end users, the banking AI chat-based user interface is the main point of access. A text area allows users to enter natural language banking questions. By examining the query to both the locally stored dataset and the uploaded KB, the system tries to find a solution. As you see in Figure 4, I requested the chat box, “How can I create a new bank account? The chatbot displays the response to the query after verifying with the provided KB file.

You can see that I asked a question that isn’t in the KB in Figure 5 below. It enables the opportunity to report the knowledge gap and verifies with KB. We can determine the precise information the user requested from this reporting area.



Banking AI

Ask a banking question:

How can I check the bank insurance policies?

Sorry... I don't have information on that topic in my current database.

▼ 🚩 Help me learn: Report a Knowledge Gap

Query: How can I check the bank insurance policies?

What specific information should the system have provided?

How can I check the bank insurance policies ?

Press Ctrl+Enter to apply

Submit to System Trainer

Figure 5. Knowledge gap detection and feedback module

The knowledge gap detection module is activated when the system cannot respond due to missing data. The system asks the user to explain the missing information rather than ending the conversation. Interacting with the user to describe the missing information is seen in Figure 5 above. This input is recorded as a knowledge gap and saved with metadata like query context and timestamp.

By converting user discontent into useful information, this module applies stakeholder-driven needs elicitation. The system trainer can analyze and handle documented knowledge gaps using the system trainer dashboard. The original user query and the information that was found to be missing are shown for each pending gap.

```

ID: REQ-184724
Query: insurance policy
Spec: The system shall provide details on insurance policy
-----
ID: REQ-035618
Query: what are the daily limits on daily withdrawals or transfers
Spec: The system shall provide details on what are the daily limits on daily withdrawals or transfers
-----
ID: REQ-035706
Query: do you have mobile app and online banking
Spec: The system shall provide details on do you have mobile app and online banking
-----
ID: REQ-035741
Query: what are fraud protection services
Spec: The system shall provide details on what are fraud protection services
-----
ID: REQ-035829
Query: how do i report a suspicious transaction or fraud
Spec: The system shall provide details on how do i report a suspicious transaction or fraud
-----
ID: REQ-055548
Query: insurance policy
Spec: The system shall provide details on insurance policy
-----

```

Figure 6. Requirement specification text

A FR can be entered directly into the SRS document via the interface once it has been verified. As previously stated, every entry in Figure 6 corresponds to a distinct need identity and is retained permanently. This system will provide traceability between across user queries, identify any gaps and match FR and NFR needs with RE best practices.

The proposed approach is based on RAG, which unites prompt-driven generation with semantic retrieval. In order to enable rapid and efficient resemblance search operations, the dataset's current responses are initially encoded into vector embeddings using Sentence Transformers (all-MiniLM-L6-v2), which is based on the Sentence-BERT approach for semantic sentence representation (Reimers & Gurevych, 2019). These embeddings are then added to an FAISS index, which supports efficient similarity search over dense vector representations (Douze et al., 2025). The requirement generation component was implemented using the Hugging Face Transformers pipeline with the FLAN-T5 text-to-text generation model. The embedding component used Sentence Transformer all-MiniLM-L6-v2 to encode banking responses into vector embeddings. These embeddings were indexed using FAISS IndexFlatL2 for semantic similarity search.

Query	Functional Requirement
How can I open a new bank account?	The system shall have the following features in order to enable me to open a new bank account.
What do I need to provide to open a bank account?	The system shall be designed to automatically open a new account and automatically close the account without the user providing additional information.
How can I check my account balance?	The system shall support customers to view their account balance using the following method:
How do I transfer money between accounts?	The system shall allow me to close my bank account
Can I set up direct deposit for my paycheck?	The system shall allow the user to set up direct deposit for their paycheck.
How do I report a lost or stolen debit card?	The system shall provide a user interface for a user to
What should I do if I forget my online banking password?	The system shall support the following steps for recovery of the forgotten password.
How can I apply for a loan?	The system shall allow me to apply for a loan from our website or from a branch.
What are your current interest rates for savings accounts?	The system shall be capable of determining the interest rate of savings accounts.
How do I close my bank account?	The system shall accept a customer's bank account number as a valid account number and shall be able to close the account with that number.
Can I update my personal information online?	The system shall allow me to update my personal information such as address or phone number by logging into my online banking account.
What is the process to dispute a transaction?	The system shall provide the user with the following information:

Figure 7. Extracted true functional requirements

During generation, the retrieved context and the user query were inserted into a fixed prompt template that began with generate a clear software requirement using the format "The system shall...". The model was used without fine-tuning. Default Hugging Face pipeline parameters were used where explicit values were not assigned.

The user search query and the gathered circumstances are put together in the subsequent step to provide a structured prompt that instructs the language model in order to produce a query. By imposing a preset syntax that begins with "The system shall", the prompt guarantees consistency in output. The text-to-text generation model depicted in Figure 7 is then used to create FR in an attainable, logical and tested manner. Most significantly, the categorisation element is implicit: rather than formally specifying criteria as FR or NFR. The structured prompting technique ensures that all created outputs are automatically categorised as functional needs. By combining retrieval, creation and template support, the system can transform incoherent user enquiries into conventional and well-organised requirement statements.

The goal of the standard classification metrics is to provide a reference point for the system's accuracy by objectively assessing the basic BERT classifier's performance in distinguishing between FR and NFR. It will be tested on a predetermined, labelled test set after being assessed by the classifier. For FR and NFR classification, the following metrics that will be reported: accuracy, recall and F1-score. The manually annotated ground truth labels and the system-generated labels used to assess the performance of FR and NFR classification are shown in Table 7.

Table 7. FR/NFR classification ground truth and model predictions

ID	Query	Generated Requirement	True Label	Predicted Label
1	How can I open a new bank account?	The system shall have the following features in order to enable me to open a new bank account.	FR	FR
2	What do I need to provide to open a bank account?	The system shall be designed to automatically open a new account and close the account without additional information.	NFR	FR
3	How can I check my account balance?	The system shall support customers to view their account balance using the following method.	FR	FR
4	How do I transfer money between accounts?	The system shall allow me to close my bank account.	FR	FR
5	Can I set up direct deposit for my paycheck?	The system shall allow the user to set up direct deposit for their paycheck.	FR	FR
6	How do I report a lost or stolen debit card?	The system shall provide a user interface for a user.	FR	FR
7	What should I do if I forget my online banking password?	The system shall support the following steps for recovery of the forgotten password.	FR	FR
8	How can I apply for a loan?	The system shall allow me to apply for a loan from our website or from a branch.	FR	FR
9	What are your current interest rates for savings accounts?	The system shall be capable of determining the interest rate of savings accounts.	NFR	FR
10	How do I close my bank account?	The system shall accept a customer's bank account number and close the account.	FR	FR
11	Can I update my personal information online?	The system shall allow the user to update personal information by logging into online banking.	FR	FR
12	What is the process to dispute a transaction?	The system shall provide the user with the following information.	NFR	FR

To evaluate the proposed system's performance, a confusion matrix is constructed, with FR as the positive class. For the confusion matrix, there are 12 samples, 9 True FR, 3 True NFR, 12 Predicted FR and 0 Predicted NFR are shown in Table 8. The TP represents correctly predicted FR, FP represents NFR incorrectly predicted as FR, FN represents FR incorrectly predicted as NFR and TN represents correctly predicted NFR. In this study, FR is considered as positive class and NFR the negative class.

Table 8. Confusion matrix (FR as positive class)

Actual \ Predicted	FR	NFR
FR	TP = 9	FN = 0
NFR	FP = 3	TN = 0

This emphasizes the current difficulty in discerning across FR and NFR specifications, but it also shows good performance in identifying FR. The displayed labels are contrasted with manually input standard reference following the system's generation and classification of the requirement specifications. These results are used to generate evaluation metrics and create a confusion matrix. Consequently, the formula is employed in the final assessment phase of the proposed approach's overall system, following the FR/NFR classification phase, which point the system has already provided measurable findings.

This indicates how challenging it is now to make the distinction between FR and NFR requirements and reveals how well the algorithm identifies FR. Some NFR occurrences are still falsely classified as FR, which affects the overall balance of classification performance, even if the suggested technique effectively covers most FR.

The classification performance is evaluated using standard metrics, including precision, recall, F1-score and accuracy which are widely used in ML and classification tasks (Sokolova & Lapalme, 2009).

In Eq. (1), the system's precision is 75% when making a functional need prediction. This shows that even when the approach does a good job of recognizing FRs, a few false-positive situations are still erroneously identified as NFRs.

$$Precision = \frac{TP}{TP + FP} = \frac{9}{9 + 3} = 0.75. \quad (1)$$

The system successfully identified all FRs in Eq. (2). This result shows how well the recommended method captures FRs and prevents vital operational needs from being overlooked during the requirement generation.

$$Recall = \frac{TP}{TP + FN} = \frac{9}{9 + 0} = 1.00. \quad (2)$$

In Eq. (3), the F1-score shows a strong balance between correctness and completeness for FR detection. This shows that the system maintains a high degree of wholeness while producing somewhat precise functional needs.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} = 2 * \frac{0.75 * 1.00}{0.75 + 1.00} = 0.857. \quad (3)$$

The system correctly classified 75% of all requirements in Eq. (4). The result also shows that some NFRs are still improperly categorized as FRs, which weakens the system-wide balance in requirement sort classification, even though this still yields sufficient performance overall.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{9 + 0}{12} = 0.75. \quad (4)$$

In Figure 8 illustrates the performance of the proposed system for classifying FR. It illustrates the values of precision, recall, F1-score and overall accuracy, allowing a direct comparison of the classification performance.

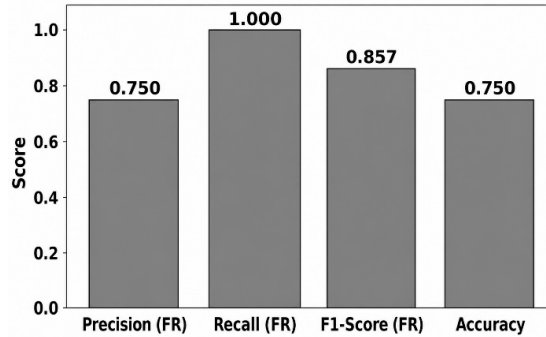


Figure 8. FR Classification Performance Metrics

All True FR were correctly detected, according to the findings, which demonstrate recall value of (1.00). False positives, in which some NFRs were mistakenly classified as functional requirements, are indicated by precision and accuracy scores of 0.75. The total F1-score of 0.857 indicates a good balance between recall and accuracy, supporting the efficiency of the suggested strategy while highlighting areas that still need work.

5. Discussion

The structure of this research emerged by applying a feedback-based SRS approach with a traditional RAG process. Nevertheless, depending on the domain level of complexity, availability of data, and organisational needs, these systems were unable to stay entirely traditional in real-world SE scenarios and rather evolve into blended systems. The purpose of the research was to improve the SRS processes by applying AI-based development. Further research can improve the approach by using specialised optimisation methods, composite classification approaches, or more complicated extraction methods.

This case study results show that in an interactive system domain, the proposed AI-enabled SRS approach can potentially promote automated and reactive requirements definition. The approach proved that domain-specific knowledge retrieval combined with LLM based generation can generate structured, context aware, discovered required outputs using the banking chatbot dataset. One of the primary limitations of standalone LLM-based requirements generations approaches was promptly overcome through the establishment of a specialised KB, providing enhanced semantic grounding and reduced ambiguity in customer queries (Niu et al., 2024; Zhang & Zhang, 2025).

In this regard, the results lend confidence to the assumption that retrieval-based generation is more effective than simple generating simple standardised for defining requirements tasks particularly in domains where contextual significance and accuracy are vital. The presence of a feedback-based persistent learning loop is a significant feature of the provided approach. In comparison to traditional RE methods, the recommended system retains either partial or defective chatbot replies as valid cues for system enhancement. During the customer interactions, the KB are noted, examined and converted into new requirement inputs.

The KB and chatbot update cycle then utilises these in a structured manner. As a result of this model is especially relevant to AI-based systems because the needs are flexible and may vary over time due to altering customer expectations, limitations or domain understanding. The recommended approach goes toward an adaptive RE process rather than generating requirements.

A significant limitation is that the empirical evaluation is based on the domain-specific case study with a limited banking chatbot dataset. Therefore, limitations associated with the chosen domain, the limited size of KB, and the limited area of the prototype environment may be reflected in the results. Further, while NFR need recognition remains its inception, the existing system mainly shows the success rate of FR generation.

Overall, what came out of this research proven that the suggested RAG based AI-enabled approach offers an enhancement in the SRS. The results obtained reveal that contextual grounding, ambiguity reduction and more adaptive requirements specification are capable of being unifying domain-specific information extraction, LLM-based requirements generation and feedback-based continuous learning. The study shows that retrieval-based and constantly updated AI help may greatly enhance the SRS process, although the present implementation is constrained by dataset size, domain scope and poor NFR handling. These results advance our understanding of how AI may be used to assist an expanding, more robust engineering process, in addition to outsourcing requirement creation processes.

6. Conclusions and future works

Analysis of related activities confirms that the dynamic, data-driven and context-dependent character of AI-based systems can be hard for traditional requirements engineering methodologies to handle. While current AI-enabled supported methods increase automation they usually lack domain base, flexibility and the potential for constant learning.

This study proposed an AI-enabled SRS approach based on RAG has been proposed. To provide more contextual and traceable needs, the system combines domain-specific information retrieval, LLM-based demand generation and a feedback-driven learning loop. A banking chatbot case study that includes Sentence-BERT embeddings, FAISS retrieval and a Streamlit interface was used to create a prototype.

The recommended approach generates and uncovers functional needs with consistent performance criteria according to experimental analysis. The system is capable of being consistently gather substantial functional specifications from customer queries as proven by its high level of recall in detecting FRs. This is particularly helpful in the initial phases of requirements elicitation, as overlooking important needs can adversely affect subsequent phases of design and execution. Apart from that, the findings show that certain non-functional needs are incorrectly labelled as FR. This suggests that the balanced FR/ NFR classification is does not prove as effective at extracting functional needs that the current version of the approach. However, the overall findings support the idea that combining RAG with a feedback-oriented learning cycle enhances the quality of requirements, lessens uncertainty, and reinforces the contextual foundation of the SRS process.

The findings of the paper shows that RAG offers a workable and scalable approach for modern AI-enabled SRS. The recommended approach provides a significant step towards more reliable, modular and semantically based RE for intelligent software systems by mixing contextual information retrieval, structured LLM-based generation and active system building through monitoring. The paper primary novelty is the contributes to a scalable and adaptive solution that integrates technical precision, setting the benchmark for future advances in the engineering of AI requirements.

Future work should address the limitations identified in this study. First, expand the dataset with larger, increase robustness and lessen domain-specific bias and multi-domain needs requirements to the dataset. Second, clearer FR/NFR annotation rules and a dedicated classification strategy should be developed to improve the identification of NFRs. Third, implement more effective retrieval, prompt engineering and validation methods to further enhance accuracy and lower classification errors. Finally, the proposed approach should be tested in additional domains beyond banking chatbots to evaluate its generalizability.

Acknowledgements

The authors would like to express sincere gratitude to Prof. Dr. Diana Kalibastienė for her valuable guidance and constructive comments.

Conflict of interest

The authors declare no conflict of interest.

References

- Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2(November), Article 100006. <https://doi.org/10.1016/j.mlwa.2020.100006>
- Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., & Grundy, J. (2023). Requirements practices and gaps when engineering human-centered Artificial Intelligence systems. *Applied Soft Computing*, 143, Article 110421. <https://doi.org/10.1016/j.asoc.2023.110421>
- Borg, M., & Borg, M. (2024). Requirements engineering and large language models: Insights from a panel. *IEEE Software*, 41(2), 6–10. <https://doi.org/10.1109/MS.2023.3339934>
- Cheng, H., Husen, J. H., Lu, Y., Racharak, T., Yoshioka, N., Ubayashi, N., & Washizaki, H. (2026). Generative AI for requirements engineering: A systematic literature review. *Software – Practice and Experience*, 56(2), 141–170. <https://doi.org/10.1002/spe.70029>
- Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P. E., Lomeli, M., Hosseini, L., & Jégou, H. (2025). the Faiss Library. *IEEE Transactions on Big Data*, 12(2), 346–361. <https://doi.org/10.1109/TBDDATA.2025.3618474>
- Fan, W. Q., Ding, Y. J., Ning, L. B., Wang, S. J., Li, H. Y., Yin, D. W., Chua, T. S., & Li, Q. (2024). A survey on RAG meeting LLMs: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024* (pp. 6491–6501). Association for Computing Machinery. <https://doi.org/10.1145/3637528.3671470>
- Ferrari, A. (2023). Artificial intelligence in engineering and society: Blue skies, black holes, and the job of requirements engineers (keynote). In K. Schneider, F. Dalpiaz, & J. Horkoff (Eds.), *2023 IEEE 31st International Requirements Engineering Conference Workshops, REW* (pp. 67–67). IEEE. <https://doi.org/10.1109/REW57809.2023.00018>

- Habiba, U. E., Haug, M., Bogner, J., & Wagner, S. (2024). How mature is requirements engineering for AI-based systems? A systematic mapping study on practices, challenges, and future research directions. *Requirements Engineering*, 29(4), 567–600. <https://doi.org/10.1007/s00766-024-00432-3>
- Kaur, K., & Kaur, P. (2024). The application of AI techniques in requirements classification: A systematic mapping. *Artificial Intelligence Review*, 57, Article 57. <https://doi.org/10.1007/s10462-023-10667-1>
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020, December). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*.
- Liu, K., Reddivari, S., & Reddivari, K. (2022). Artificial intelligence in software requirements engineering: State-of-the-Art. In *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science, IRI* (pp. 106–111). IEEE. <https://doi.org/10.1109/IRI54793.2022.00034>
- Maalej, W., Pham, Y. D., & Chazette, L. (2023). Tailoring requirements engineering for responsible AI. *Computer*, 56(4), 18–27. <https://doi.org/10.1109/MC.2023.3243182>
- Nalchigar, S., Yu, E., & Keshavjee, K. (2021). Modeling machine learning requirements from three perspectives: A case report from the healthcare domain. *Requirements Engineering*, 26(2), 237–254. <https://doi.org/10.1007/s00766-020-00343-z>
- Niu, C., Wu, Y., Zhu, J., Xu, S., Shum, K., Zhong, R., Song, J., & Zhang, T. (2024). RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. In L. W. Ku, A. Martins, & V. Srikumar (Eds.), *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (vol. 1, pp. 10862–10878). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2024.acl-long.585>
- Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. <https://doi.org/10.18653/v1/D19-1410>
- Siddeshwar, V., Alwidian, S., & Makrehchi, M. (2024). A systematic review of AI-enabled frameworks in requirements elicitation. *IEEE Access*, 12, 154310–154336. <https://doi.org/10.1109/ACCESS.2024.3475293>
- Sofian, H., Yunus, N. A. M., & Ahmad, R. (2022). Systematic mapping: Artificial intelligence techniques in software engineering. *IEEE Access*, 10, 51021–51040. <https://doi.org/10.1109/ACCESS.2022.3174115>
- Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4), 427–437. <https://doi.org/10.1016/j.ipm.2009.03.002>
- Taj, S., Daudpota, S. M., Imran, A. S., & Kastrati, Z. (2025). Aspect-based sentiment analysis for software requirements elicitation using fine-tuned bidirectional encoder representations from transformers and explainable artificial intelligence. *Engineering Applications of Artificial Intelligence*, 151, Article 110632. <https://doi.org/10.1016/j.engappai.2025.110632>
- Wang, R., Liu, J., Zhang, Q., Fu, C., & Hou, Y. (2024). Federated learning for feature-fusion based requirement classification. *Cluster Computing*, 27(3), 3397–3416. <https://doi.org/10.1007/s10586-023-04147-y>
- Zhang, W., & Zhang, J. (2025). Hallucination mitigation for retrieval-augmented large language models: A review. *Mathematics*, 13(5), Article 856. <https://doi.org/10.3390/math13050856>
- Zowghi, D., Bano, M., & Borg, M. (2023). What's missing in requirements engineering for responsible AI? *IEEE Software*, 40(6), 11–15. <https://doi.org/10.1109/MS.2023.3302934>