AI-SRS Studio: A Multi-Agent AI Framework for Automated Generation and Validation of Software Requirements Specifications

[1]Olaiya Folorunsho, [2]Temiloluwa E. Awaye, [3]Seye E. Akinsanya, and [4]Olubunmi O. Faboya
[1,2,3,4]Department of Computer Science, Federal University Oye-Ekiti, Oye Ekiti, Nigeria
olaiya.folorunsho@fuoye.edu.ng|awayetemiloluwa@gmail.com|seye.akinsanya@fuoye.edu.ng|olubunmi.faboya@@fuoye.edu.ng

Received:18-AUGUST-2026; Reviewed: 19-SEPT-2026; Accepted: 25-SEPT-2026
https://dx.doi.org/10.46792/fuoyejet.v10i3.1645

ORIGINAL RESEARCH

Abstract— The Software Requirement Specification (SRS) is a formally written document which includes the list of characteristics, requirements, and specifications of a software product. It is not easy to develop an SRS manually and may be ambiguous, but although the large language models like ChatGPT have excellent text generation capabilities, the majority of general-purpose tools do not provide the domain-specific logic to ensure the compliance of requirements, their completeness, consistency and traceability. In this paper, we propose the solution named AI-SRS Studio which is a web-based multi-agent artificial intelligence framework designed to support the automated generation and quality assessment of SRS and Product Requirements Document (PRD) artifacts. The Design Science Research methodology was used and the tool was built using Next.js, Groq inference API, and Supabase database. The system underwent validation using expert review along with user acceptance test by developers, designers, and project managers. Feedback results on the basis of 36 responses indicated an average 3.6/5.0 (72%), while the automated evaluation of the generated SRS document yielded an overall quality score of 81%. The evaluation scores were 77% for completeness, 76% for consistency, 89% for unambiguity, and 83% for traceability. However, the expert evaluation involved only three subject-matter experts, which may limit the generalisability of the findings. The findings indicate that the proposed multi-agent workflow can support the production of structured requirements documentation and provide an effective mechanism for preliminary requirements-quality assessment. Furthermore, the results indicate that with multi-agent and prompt engineering, fuzzy project ideas can be converted to requirements documents. This study highlights that software development teams need to integrate the multi-agent system into their existing documentation processes as a document generation accelerator instead of an entirely automated process.

Keywords— Artificial Intelligence, Multi-Agent Systems, Prompt Engineering, Requirements Engineering, Software Requirement Specification, Product Requirement Documentation.

1 INTRODUCTION

Software Requirements Specification (SRS) is an important software engineering artifact that defines the functional and non-functional requirements, characteristics, and constraints of a software product (Asif et al., 2019). SRS provides a common understanding among stakeholders, developers, designers, testers, and project managers and serves as a foundation for subsequent software development activities (Oran et al., 2021). However, developing a high-quality SRS manually can be time-consuming and challenging, particularly when requirements originate from informal, incomplete, or ambiguous descriptions of stakeholder needs (Rosado da Cruz et al., 2025). Poorly specified requirements can result in misunderstandings, rework, increased development costs, and software products that fail to satisfy user expectations (Bjarnason et al., 2011) The emergence of Large Language Models (LLMs) has created new opportunities for automating requirements engineering activities (Zadenoori et al., 2025). Generative AI tools such as ChatGPT can transform natural-language descriptions into structured requirements, user stories, acceptance criteria, and other software engineering artifacts (Marques et al., 2024; Cheng et al., 2026). These capabilities can significantly reduce the effort required to produce initial requirements documentation. However, general-purpose LLMs are primarily designed for broad language-generation tasks and do not inherently guarantee compliance with established requirements engineering principles. AI-generated requirements may therefore contain omissions, ambiguity, inconsistency, redundancy, or weak traceability (Sigloch et al., 2026). Recent studies have investigated the use of LLMs for requirements elicitation, specification, analysis, and validation (Arora et al., 2024; Hemmat et al., 2025). However, much of the existing work focuses on individual requirements engineering tasks or the generation of requirements artifacts. There remains a need for integrated AI solutions that can not only generate requirements but also systematically evaluate and refine the resulting documentation according to explicit quality criteria. This is particularly important because fluent and plausible AI-generated text does not necessarily imply that the resulting requirements are complete, consistent, unambiguous, and traceable. Multi-agent AI provides a promising approach to addressing this limitation. Instead of relying on a single AI agent to perform all requirements engineering activities, specialised agents can collaborate to generate, analyse, evaluate, and refine requirements (Sami et al., 2024). Such architecture can provide a more structured workflow and allow different aspects of requirements quality to be

Corresponding Author
Section B- ELECTRICAL/COMPUTER ENGINEERING & RELATED SCIENCES
Can be cited as:
Folorunsho O., Awaye T. E., Akinsanya S. E., and Faboya O. O. (2026). AI-SRS Studio: A Multi-Agent AI Framework for Automated Generation and Validation of Software Requirements Specifications. FUOYE Journal of Engineering and Technology (FUOYE**JET), 11(3), 435-440.**
https://dx.doi.org/10.46792/fuoyejet.v11i3.1645

© 2026 The Author(s). Published by Faculty of Engineering, Federal University Oye-Ekiti.
This is an open access article under the CC BY NC license. (https://creativecommons.org/licenses/by-nc/4.0/)

examined before the final documentation is produced. Nevertheless, the practical integration of multi-agent AI into unified requirements engineering environment remains insufficiently explored. To address this gap, this study proposes AI-SRS Studio, a web-based multi-agent AI framework for automated generation and validation of Software Requirements Specification (SRS) and Product Requirements Document (PRD) artifacts. The proposed system transforms relatively informal software project ideas into structured requirements documentation and evaluates the generated document according to four quality dimensions: completeness, consistency, unambiguity, and traceability. The system was developed using Next.js, the Groq inference API, and Supabase and was guided by the Design Science Research methodology. The developed AI-SRS Studio was evaluated through expert review and user acceptance testing involving developers, designers, and project managers. The evaluation also examined the quality of generated requirements using the proposed automated assessment mechanism. The study therefore investigates whether a multi-agent AI approach can effectively transform relatively vague software-product ideas into structured and quality-assessed requirements documentation.

The main contributions of this study are: (1) a multi-agent AI framework for automated SRS/PRD generation; (2) an integrated mechanism for evaluating requirements quality based on completeness, consistency, unambiguity, and traceability; and (3) an empirical evaluation of the developed web-based artifact through expert review, automated quality assessment, and user acceptance testing. The study contributes to the emerging field of generative AI-assisted requirements engineering by integrating requirements generation and quality assessment within a unified workflow.

2 RELATED WORKS

A SRS is a binding document and a means of communication between technical experts and business parties, making sure that the suggested architecture is consistent with the initial business goals (Wei et al., 2023). The development of traditional SRS involves manual elicitation via interviewing and workshopping, which is expertise-intensive, laborious and risky procedure because poor quality of requirements is a frequent cause of software project rework (Muhamad et al., 2023). Large Language Models generate natural language text based on huge training corpora and can be used for text generation, summarisation, and dialogue (Hadi et al., 2023). Recent research has demonstrated the application of large language models in domain-specific intelligent systems, including medical diagnosis and treatment recommendation (Isijola *et al.*, 2025), highlighting the potential of LLMs for specialised, task-oriented applications. According to Teubner et al. (2023), the efficiency of LLMs largely depends on the user's proficiency in prompt engineering, which is the practice of formulating input to get better results from it. Prompts for LLMs can be Zero-Shot and Few-Shot in-context learning, Chain-of-Thought (CoT) prompting forcing a model to demonstrate reasoning steps prior to generating the result (Wei et al., 2022), and Tree-of-Thoughts (ToT), which extends CoT into a non-linear reasoning over multiple alternative solutions (Yao et al., 2023). Manual review of the SRS is usually done, relying greatly on the intelligence of the reviewer, and being quite a tedious process, it normally results in semantic and structural mistakes going undetected (Wang et al., 2025). On the other hand, the automated evaluation system evaluates a document on the basis of quality criteria that are clearly defined and taken from the standard such as ISO/IEC/IEEE 29148: Completeness, when all entities have full CRUD coverage and edge cases are addressed; Consistency, when there are no internal conflicts in the specification; Unambiguity, when the requirement allows only one interpretation and does not contain subjective statements such as “fast” and “robust”; and Verifiability, when all requirements have enough precision in their metrics for testing.

Zhu et al. (2025) introduced ReqInOne, which is an LLM-based agent, to draft the SRS, in terms of semantic extraction, constraint validation and formal specification, thus lessening the problem of hallucination compared to one-shot generation. The study by Helfrich and Regino (2025) revealed that the creation of a specific persona and constraints for an autonomous agent can substantially increase engineering standards compliance. Cheng et al. (2026), who conducted a systematic review of using the generative AI for RE, stated that the general-purpose models lack domain-specific knowledge necessary for ensuring completeness and traceability. Kallioinen (2025) found out that developers like to have the AI capability in their workflow but not in an additional chat window, whereas Arora et al. (2024) highlighted particular prompt patterns, in particular "Output Customisation". Poorly written requirements continue to be one of the most frequent causes of project reworks; more than 87% of the requirements' documentation is written in natural language, and it is therefore prone to misunderstanding (Muhamad et al., 2023; Dar et al., 2022), and problems associated with structure in requirements' documentation have also been identified in software companies where English was not their language (Jiaying et al., 2024). While all the above progress has been made, there still lacks literature on the integration of agent-based generation of requirements with the evaluation layer within a unified tool. This is what inspired this research.

3 PROPOSED METHODOLOGY

This section presents a detailed explanation of the proposed framework used to build the AI-SRS Studio. The research adopted the Design Science Research (DSR) approach, which is an iterative methodology based on the design of the artifact and its usability evaluation with

© 2026 The Author(s). Published by Faculty of Engineering, Federal University Oye-Ekiti.
This is an open access article under the CC BY NC license. (https://creativecommons.org/licenses/by-nc/4.0/)

http://dx.doi.org/10.46792/fuoyejet.v10i3.1645 engineering.fuoye.edu.ng/journal

respect to the current manual documentation process DSR is appropriate for this study because the primary objective was to design, implement, and evaluate an information technology artifact that addresses a practical problem in software requirements engineering.

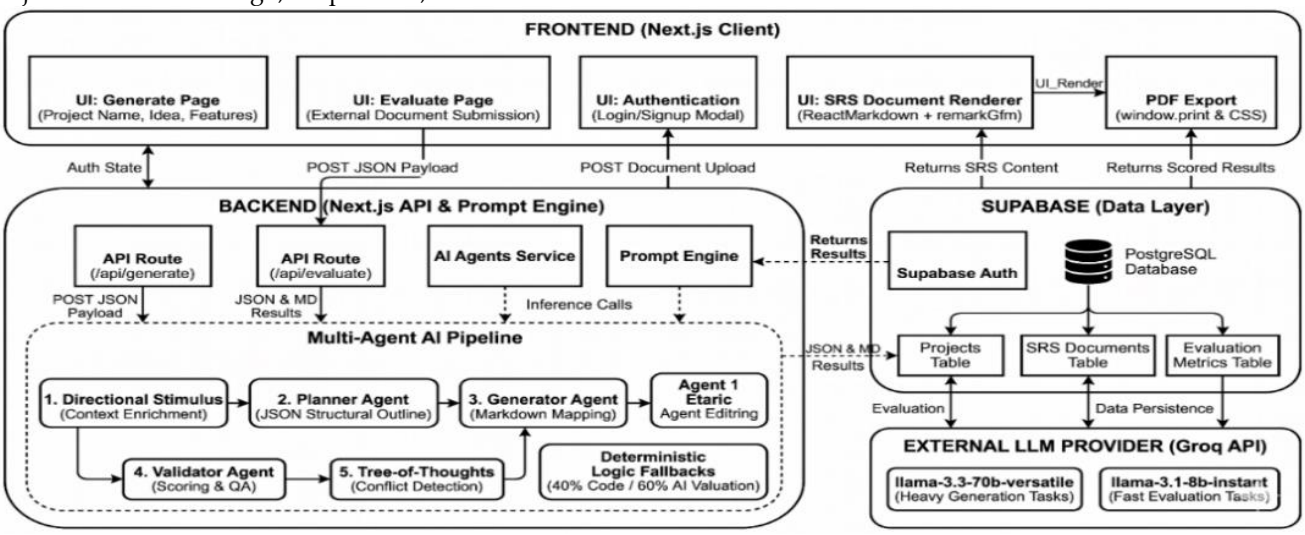

Fig. 1. System architecture of AI-SRS Studio

The AI-SRS Studio architecture presented in Figure 1 is made up of four distinct layers: Next.js front-end, Next.js API and prompt engine back-end, multi-AI agent pipeline, and data provider (Supabase and Groq API). The front-end records the ideas of the project, its target audience, the features, and technologies to be used, generates the document in ReactMarkdown, and exports it in PDF format. The front-end provides the user interface, authentication through Supabase, SRS document rendering, and PDF export functionality. The back-end has serverless API routes that validate the request and call the AI Agents Service for the pipeline orchestration. It manages communication between the frontend and AI models, including API processing, agent orchestration, and prompt engineering.

3.1 Multi-Agent Pipeline

The basic workflow consists of the Chain-of-Thought pipeline that consists of the following agents: the Planner (temperature 0.4) breaks down the enriched input idea into stakeholders and goals/constraints as a JSON blueprint; the Generator (temperature 0.5) develops the JSON blueprint into an IEEE-compliant Markdown SRS with each functional requirement linked to one stakeholder goal; and the Validator (temperature 0.2) evaluates the generated SRS draft and outputs the list of found issues. The Tree-of-Thoughts module implements the three-step ENUMERATE–EVALUATE–BACKTRACK process that finds conflicting requirement pairs; additionally, the Self-Consistency Evaluator runs three parallel reasoning paths (IEEE 29148 reviewer, security specialist, end user), taking into account only the issues reported by at least two paths, except security issues. Deterministic Heuristic Layer separately evaluates for completeness, ambiguity, atomicity, and verifiability and fuses the output with the AI-driven score with a weightage of 40:60.

The Directional Stimulus Pre-Processor is deterministic in nature and identifies fuzzy inputs using either a less-than-25-word threshold or less-than-30% coverage of 16 domain-specific signal terms. It then adds up to six domain-specific stimulus words before invoking the LLM, without incurring additional inference costs. The thresholds were determined through iterative development and preliminary testing. The 25-word threshold, 30%/16-term cutoff, and six stimulus words were adopted as practical heuristic values for identifying potentially underspecified inputs and determining when additional preprocessing was necessary. These values were selected to balance sensitivity to incomplete project descriptions with the avoidance of unnecessary preprocessing for sufficiently detailed inputs.

3.2 Module Breakdown

Backend logic is split across six modules with an acyclic dependency graph, where the API route is dependent on the service module that is dependent on the prompt modules, which in their turn are dependent solely on the prompt-engine module (Table 1).

Table 1. Backend Module Breakdown

Module	Role
API Route Handler	Validates the payload of the request and triggers the AI Agents Service
Directional Stimulus Pre-Processor	Performs deterministic augmentation of the fuzzy input prior to making an LLM call
CoT Pipeline Orchestrator	Triggers the sequence of execution of Planner, Generator, and Validator agents
Tree-of-Thoughts Conflict Detector	Lists, evaluates, and resolves conflicting requirements pairs
Self-Consistency Evaluator	Executes three parallel evaluators of external documents
Deterministic Heuristic Layer	Deterministic rule-based scoring and AI scores in 40:60 proportion

3.3 Evaluation Design

Validation of the System included the automated quantitative benchmarking as well as the qualitative User Acceptance Testing (UAT). The UAT involved 36 participants comprising developers, designers, and project managers who performed general usability testing, as well as three Subject Matter Experts (SMEs) who confirmed the presence of hallucinations or

© 2026 The Author(s). Published by Faculty of Engineering, Federal University Oye-Ekiti.
This is an open access article under the CC BY NC license. (https://creativecommons.org/licenses/by-nc/4.0/)

http://dx.doi.org/10.46792/fuoyejet.v10i3.1645 engineering.fuoye.edu.ng/journal

architectural conflicts in the requirements that have been generated.

4 RESULTS AND DISCUSSION

The AI-SRS Studio was developed using Next.js 14 (App Router) and TypeScript while using Tailwind CSS for the application's styling. The documentation process utilised llama-3.3-70b-versatile while the evaluation passes were conducted using llama-3.1-8b-instant, both accessed via the Groq API while Supabase handled the PostgresSQL data storage and authentication processes and hosting on Vercel (Table 2).

Table 2. Core Technology Stack

Technology	Role
Next.js 14	Server-side routes keep the API key off the client
TypeScript	Type-safe multi-agent pipeline contracts
Groq API	Low-latency inference for all agents
Supabase	Auth, persistence, Row-Level Security

4.1 Automated Quality Metrics

Four deterministic heuristics were integrated using the scoring of the Validator Agent (40:60 ratio): Completeness (ratio of presence of six keywords in sections), Ambiguity (count of ten banned subjective keywords), Atomicity for Consistency (ratio of non-compound requirement sentences), and Verifiability for Traceability (LLM estimated percentage of testable requirements). The insistence on only JSON outputs decreased the number of JSON parse errors by about 12% to less than 1%, demonstrating improved reliability in the generation of structured JSON outputs.

4.2 User Acceptance and Document Quality

The feedback feature of the platform depicted in Figure 2.

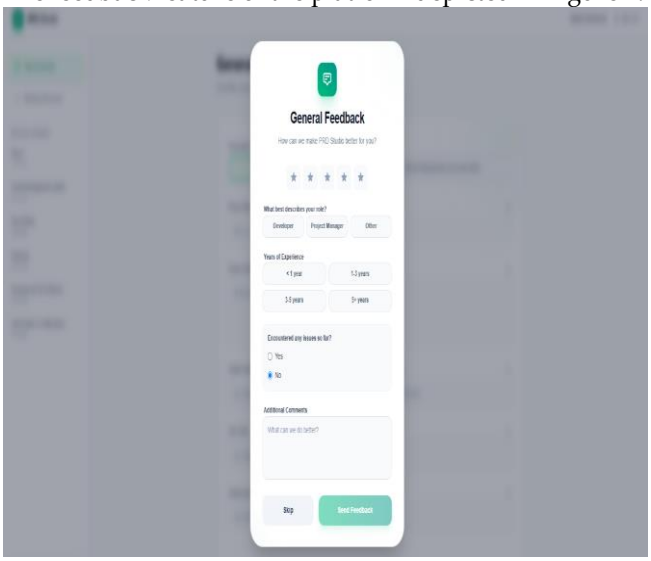

Fig. 2. Feedback Form Integrated Into the Platform

obtained a total of 36 answers from developers, project managers and other personnel and received a mean satisfaction score of 3.6/5.0 and the average rating shown in Figure 3. The feedback feature of the platform, depicted in Figure 2, was used to conduct User Acceptance Testing (UAT) with 36 respondents comprising developers, designers, and project managers. The participants evaluated the usability, usefulness, and overall acceptance of the proposed AI-SRS Studio framework. The results, presented in Figure 3, show a mean satisfaction score of 3.6/5.0 (72%), indicating a generally positive level of user acceptance

Fig. 3. Average Feedback Rating from Users

The quality assessment of generated documents through automated evaluation depicted in (Table 3) had an average of 81%, comprising of Completeness 77%, Consistency 76%, Unambiguity 89% and Traceability 83%, showing that the process was highly effective in removing ambiguity and relatively ineffective in ensuring consistency.

Table 3. Average Document Quality Scores

Dimension	Score (%)
Completeness	77
Consistency	76
Unambiguity	89
Traceability	83
Overall	81

4.3 Discussion

The results suggest that decomposing the SRS creation process into the Planner-Generator-Validator pipeline, combined with the additional ToT conflict checker and the majority-vote self-consistency evaluation module, improves the quality of the generated requirements compared with the single-prompt monolithic approach. The final Completeness score of 77% represents a 32-percentage-point improvement over the 45% monolithic baseline reported in the baseline experiment, corresponding to approximately 1.7 times the baseline score. As shown in Table 3, Unambiguity recorded the highest score (89%), followed by Traceability (83%), while Completeness (77%) and Consistency (76%) recorded relatively lower scores. This suggests that although the system produces relatively clear and traceable requirements, maintaining completeness and consistency across the generated SRS remains an area for further improvement.

The average UAT score of 3.6/5.0, together with the overall document quality score of 81%, indicates that the

© 2026 The Author(s). Published by Faculty of Engineering, Federal University Oye-Ekiti.
This is an open access article under the CC BY NC license. (https://creativecommons.org/licenses/by-nc/4.0/)

http://dx.doi.org/10.46792/fuoyejet.v10i3.1645 engineering.fuoye.edu.ng/journal

system can support the acceleration of SRS draft production while retaining the need for human review and validation. This finding is consistent with Kallioinen's (2025) work on human-in-the-loop embedded AI solutions, which emphasises the complementary role of AI in supporting rather than completely replacing human involvement.

4.4 Implementation Constraints

The free tier of Groq during the development process is constrained by 30 requests and 6,000 tokens per minute for llama-3.3-70b-versatile, and 30 requests and 20,000 tokens per minute for llama-3.1-8b-instant (Table 4). The cost of generating one request is about 21,300 tokens, which is near the limit for the 70B model per minute, thus influencing two architectural design choices: the CoT pipeline executes sequentially rather than parallel and the Self-Consistency module is explicitly implemented based on the smaller 8B model since three concurrent evaluation requests should fit the rate. A try/catch mechanism providing a neutral 0.5 verifiability score in case of an error in Groq API call guarantees resilience to a possible failure of API.

Table 4. Groq Free-tier API Rate Limit

Model	RPM	TPM Limit
llama-3.3-70b-versatile	30	6,000
llama-3.1-8b-instant	30	20,000

As shown in Table 4, the Groq free-tier API imposed both request and token limits on the models used by the system. The 70B model was subject to a lower token limit of 6,000 TPM, while the 8B model allowed up to 20,000 TPM. These constraints affected the execution strategy and required the agents to be executed sequentially rather than in parallel, potentially increasing the overall processing time. Five rounds of prompt design were conducted. The initial monolithic prompt was rejected after achieving approximately 45% completeness. The two-step Planner-Generator pipeline improved the structure of the generated requirements but continued to produce contradictions between functional and non-functional requirements. The three-step Planner-Generator-Validator pipeline substantially reduced ambiguities but did not fully resolve contradictions between requirements. The subsequent addition of the Tree-of-Thoughts Conflict Detector addressed these contradictions but required an additional API call. Finally, the Self-Consistency Evaluation module was incorporated using three persona-based evaluations and majority voting to further improve the reliability of the generated requirements.

4.5 Challenges, Limitations, And Scalability Considerations

Despite the satisfactory performance of the proposed system, several limitations were identified. The Groq free-tier API imposed rate and token restrictions, requiring sequential rather than parallel agent execution and potentially increasing processing time. Although the Validator and Self-Consistency modules support inconsistency and conflict detection, some cross-cutting issues may still require additional validation or human review. The expert evaluation was also limited to three subject-matter experts, reducing the generalisability of findings related to hallucination and conflict detection; future studies should therefore involve larger and more diverse expert panels. Furthermore, the use of an internal dataset limits generalisability across domains, highlighting the need for evaluation with larger and more diverse datasets. From a scalability perspective, supporting more users and concurrent queries would require increased computational capacity and improved backend and API management to maintain system responsiveness and reliability.

5 CONCLUSIONS AND FUTURE WORK

The experiment proposed, developed and assessed AI-SRS Studio, a multi-agent artificial intelligence platform that would develop and assess Software Requirement Specification documentation. Through a combination of a linear Chain-of-Thought pipeline, Tree-of-Thoughts conflict detection, a three-branch Self-Consistency assessor and deterministic heuristics based assessment, the platform would take a general idea of the project and convert it into the form of IEEE/ISO conforming documentation, as well as provide a quantitative assessment of the quality of the document generated. The experiment results have proven that the average score for the generated documents was 81%, while the user satisfaction was 3.6/5.0 according to 36 respondents' feedback.

Future work could focus on enhancing the system through multi-agent conflict resolution, where multiple AI agents collaboratively identify and resolve inconsistencies in requirements; bi-directional requirements traceability, linking requirements to generated source-code templates and test cases; and real-time multimodal requirements elicitation, enabling the system to process stakeholder discussions, audio, video, and whiteboard diagrams and automatically transform the captured information into structured IEEE-compliant requirements documents.

REFERENCES

Arora, C., Grundy, J., & Abdelrazek, M. (2024). Advancing requirements engineering through generative ai: Assessing the role of llms. In *Generative AI for Effective Software Development* (pp. 129-148). Cham: Springer Nature Switzerland.

Asif, M., Ali, I., Malik, M. S. A., Chaudary, M. H., Tayyaba, S., & Mahmood, M. T. (2019). Annotation of software requirements specification (srs), extractions of nonfunctional requirements, and measurement of their tradeoff. *IEEE access*, *7*, 36164-36176.

Bjarnason, E., Wnuk, K., & Regnell, B. (2011, August). Requirements are slipping through the gaps—A case study on causes & effects of communication gaps in large-scale software development.

© 2026 The Author(s). Published by Faculty of Engineering, Federal University Oye-Ekiti.

This is an open access article under the CC BY NC license. (https://creativecommons.org/licenses/by-nc/4.0/)

In *2011 IEEE 19th international requirements engineering conference* (pp. 37-46). IEEE.

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.

Dar, H. S., Imtiaz, S., & Lali, M. I. (2022). Reducing requirements ambiguity via gamification: comparison with traditional techniques. *Computational Intelligence and Neuroscience*, *2022*(1), 3183411.

Hadi, M. U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M. B., ... & Mirjalili, S. (2023). Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea preprints*, *1*(3), 1-26.

Helfrich, J., & Regino, E. (2025). Using AI Agents in Software Requirements. https://doi.org/10.20944/preprints202510.2470.v1

Hemmat, A., Sharbaf, M., Kolahdouz-Rahimi, S., Lano, K., & Tehrani, S. Y. (2025). Research directions for using LLM in software requirement engineering: A systematic review. *Frontiers in Computer Science*, *7*, 1519437.

Isijola, A. O., Ogude, U. C., Akinsola, D. M., Isiakpona, O. I., & Asefon, M. P. (2024). Medical Diagnosis and Treatment Recommendations (Doc-Bot) using Large Language Models. *FUOYE Journal of Engineering and Technology*, 9(4), 609-614.

Jiaying, H., Yap, N. K., Osman, M. H., & Hassan, S. A. (2024). Issues in Chinese Requirements Specifications: Insights from Survey Data and Static Analysis. *JOIV: International Journal on Informatics Visualization*, *8*(4), 2273-2279.

Kallioinen, K. (2025). User study on potential LLM based Software Requirement Specification tool for Product Owners *[Master's thesis, Aalto University]*.

Marques, N., Silva, R. R., & Bernardino, J. (2024). Using chatgpt in software requirements engineering: A comprehensive review. *Future Internet*, *16*(6), 180.

Muhamad, F. N. J., Ab Hamid, S. H., Subramaniam, H., Abdul Rashid, R., & Fahmi, F. (2023). Fault-prone software requirements specification detection using ensemble learning for edge/cloud applications. Applied Sciences, 13(14), 8368.

Oran, A. C., Santos, G., Gadelha, B., & Conte, T. (2021). A framework for evaluating and improving requirements specifications based on the developers and testers perspective. *Requirements Engineering*, *26*(4), 481-508.

Rosado da Cruz, A. M., & Cruz, E. F. (2025). Machine learning techniques for requirements engineering: A comprehensive literature review. *Software*, *4*(3), 14.

Sami, M. A., Waseem, M., Zhang, Z., Rasheed, Z., Systä, K., & Abrahamsson, P. (2024). AI based multiagent approach for requirements elicitation and analysis. *arXiv preprint arXiv:2409.00038*.

Sigloch, P., & Benzmüller, C. (2026). Neuro-Symbolic Verification of LLM Outputs for Data-Sensitive Domains (extended preprint). *arXiv preprint arXiv:2605.26942*.

Teubner, T., Flath, C. M., Weinhardt, C., Van Der Aalst, W., & Hinz, O. (2023). Welcome to the Era of ChatGPT et al. *Business & information systems engineering*, *65*(2), 95-101.

Wang, K., Li, T., Zhang, X., Wang, C., Sun, W., Liu, Y., ... & Shi, B. (2025). Software development life cycle perspective: A survey of benchmarks for code large language models and agents. *arXiv preprint arXiv:2505.05283*.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, *35*, 24824-24837.

Wei, X., Wang, Z., & Yang, S. (2023). An automatic generation and verification method of software requirements specification. *Electronics*, *12*(12), 2734.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, *36*, 11809-11822.

Zadenoori, M. A., Dąbrowski, J., Alhoshan, W., Zhao, L., & Ferrari, A. (2025). Large language models (llms) for requirements engineering (re): A systematic literature review. *arXiv preprint arXiv:2509.11446*.

Zhu, T., Cordeiro, L. C., & Sun, Y. (2025, September). ReqInOne: A Large Language Model-Based Agent for Software Requirements Specification Generation. In *2025 IEEE 33rd International Requirements Engineering Conference (RE)* (pp. 449-457). IEEE.

© 2026 The Author(s). Published by Faculty of Engineering, Federal University Oye-Ekiti.
This is an open access article under the CC BY NC license. (https://creativecommons.org/licenses/by-nc/4.0/)