© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716529

Requirement Gathering Using Agentic AI

PALOMI GAWLI[1], DEEPIKA SIDRAL[2], YASH SHINDE[3], SABURI NIKAM[4], MADHAV JAGTAP[5], TUSHAR GHORPADE[6]

[1, 2, 3, 4, 5, 6] *Department of Artificial Intelligence and Data Science, Vishwakarma Institute of Technology, Pune, India.*

Abstract- ***The process of creating a Software Requirements Specification (SRS) can be very laborious, repetitive and contain a great deal of human error during the development of software. This initiative introduces the Auto SRS generator, which is an agent based AI assisted system, enabling the automatic creation of the SRS documents from start to finish. The agent-based solution based upon intelligent agents that not only understand written descriptions of projects, but also interact with structured prompting, have the ability to analyse user input and produce accurate and extensive requirement specifications according to standard IEEE SRS guidelines. The system accepts user provided basic project information and produces a complete SRS Document by expanding it into numerous well-organized sections of Functional Requirements, Non-Functional Requirements, System Models, Use Case Descriptions and Constraints. The Auto SRS Generator reduces the effort to create a manual document, improves the quality of the document and helps software developers create a clearer, more consistent format. Additionally, the Auto SRS Generator provides information in Real-time, meaning less human involvement and consequently the generation of higher quality documentation. The experimental evaluation demonstrated that the auto-generated SRS Document was well-structured, clear, and could produce a SRS Document faster than typically done. In addition, this work has demonstrated various components of the System Architecture and Workflow and described how future work will be aimed toward facilitating the adaptation of this Tool for Academic, Industrial and Large-Scale Software Engineering Projects.***

Index Terms- ***— Auto SRS Generator, Agentic AI, Software Requirements Specification, Requirement Automation, IEEE SRS, NLP, Software Engineering, Intelligent Agents,***

I. INTRODUCTION

Writing an SRS is still a slow and messy process for most software projects today. Teams largely rely on back-and-forth messages, scattered documents, and endless revisions to finalize the requirements. This not only wastes time but also leads to misunderstandings, missing details, and misaligned expectations between clients, managers, and developers. Requirement management could get very painful as projects grow bigger. The Auto SRS Generator with agentic AI is a significant part of the solution to that problem. Instead of rewriting the same text over and over again or maintaining different document variants, you will be able to create and update everything in one place. Where needed, AI agents would suggest better phrasing, point out confusing areas, and help shape a clear, complete SRS instead of doing the usual heavy manual effort.

Besides, it is a very powerful tool to make collaboration more effective. They all work in the same environment: clients can write a first draft, managers can adjust as things change, and developers may make necessary clarifications on the spot rather than at some meeting. The system saves every chat, change, and summary, so nothing gets lost and the whole team is updated.

The main objectives of the Auto SRS Generator include:

1. Reducing the time spent on requirement gathering by offering AI-driven suggestions and structured SRS templates.
2. Improving clarity and communication through built-in Q&A and shared discussion spaces.
3. Ensuring accuracy and consistency in the SRS as it moves from client to manager to developer.
4. Creating a centralized environment where the entire requirement process from drafting to revisions to approval happens without confusion.

“This paper takes a closer look at the overall system architecture, how users move through the workflow, how AI features are built in, and the key challenges

the team encountered during development.This demonstrates how agentic AI can actually change how teams are gathering and refining requirements, away from old, time-consuming methods. And there is still plenty of room to grow-future upgrades can be made to the system Stronger, even easier to use, and something more teams will use. would want to adopt.

II. LITERATURE SURVEY

size and speed projects, there has been a pronounced shift towards the Automation of requirements gathering: With newly advanced models of artificial intelligence, collaborative platforms, and intelligent Automation, therefore, has researchers around the world finding ways in which of fast-tracking SRS creation in order to make it clear and with less need for repeated meetings and manual documentation. A lot every effort has been made to avoid confusion, improvement of communications among teams, reducing time needed to transform raw ideas into structured requirements.

Progress on Automating the Writing of Software Requirements Specifications have been pursued with zeal by developed by researchers over the last few years, most of it has come along with large language models and multi-agent AI systems. Krishna was one of those early voices that Showed how LLMs can accelerate the drafting phase of Analysts, particularly in projects requiring repetitive requirements of patterns are involved Wu et al. [2] provided a major advance in their paper 'On the Autogeneration of AI-Agents' by exploring the potential for many AI agents to be able to communicate with each other, coordinate their activities, and produce more polished results as a result of the recent growth of interest in SRS. As this interest has grown, members of the research community have begun examining how the LLM technology might be integrated into the commercial processes of Requirements Engineering. Hemmat et al. [3] have examined several crucial issues and opportunities presented by the usage of LLMs for the drafting and refinement of requirements. At the same time, Ali et al. [4] published a comprehensive survey on Agentic AI Architectures wherein multiple Agentic AI agents can cooperate to accomplish complex tasks by simplifying the task into manageable components.

The above statements are true, especially in the area of automated systems where the generation of SRS documents occurs through extraction, restructuring, validation, and formatting. Many recent research works have been published that have addressed areas of automation. Sharma and Kaur [5] have looked into early attempts to utilise AI technology to automate the generation of SRSs, however, they have noted that while LLMs are capable of easily generating clear human-readable sentences, they find producing clear, coherent, and technically accurate sentences difficult when dealing with domains that have highly technical languages. Similarly, Singh et al. [6] have echoed the same sentiments regarding the difficulties experienced when using LLMs to generate clearly defined technical descriptions from natural languages; however, they have also pointed out that the use of LLMs in automated SRS systems can ultimately save time for analysts by greatly reducing the amount of duplicate information they have to rewrite. Zhang and Liu [7] have gone a step further and investigated the potential to take the technical requirements created by LLMs to map directly into system designs, thereby creating an end-to-end solution from natural language descriptions to code. The work of Agarwal and Jain [8] has also been useful in identifying how structured prompts can help lead LLMs in deducing formal requirements when creating technical descriptions of deep technical topics. Ahmad's research [9] has taken a more comprehensive approach to identify the requirements engineering process itself and to define the requirements of an automated SRS tool, highlighting the need for clarity and traceability—all of which need to be carefully managed within an automated SRS tool. According to the research by Habiba et al. [10] in their article by the same name, LLM's achievements (and the ability to fulfil their purpose) have not yet reached a mature stage. Although the current LLM techniques may appear to have potential, the authors indicate that they still need more definitive validation at several levels.

Overall, these studies strongly suggest the realisation of automated SRS generation from LMs, Agent architectures and structured workflows. While many hurdles will remain, including the accuracy, domain knowledge and validation of generated SRSs, the trend will be towards developing AI tools that

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716529

provide intelligent interaction with users and are equipped with numerous agents to interface with technicians in the development of an automated SRS process.

III. PROPOSED SYSTEM

The aim of the system is to make the entire SRS process easy and organized. Instead of having to share documents or juggling their versions, the client, manager, and the developer will collaborate in one space. They can create the SRS here, update it, discuss doubts, and track all the decisions. The idea is to keep everybody on the same page and ensure the SRS remains accurate right from the very beginning to the end.

Describes the system that contains three kinds of users with three different functions.

a. The Client Section
Above is responsible for creating an SRS by using either automatic creation or manual input, reviewing the created information with the automated suggestion tool, making any
clarification to avoid misunderstandings/questions, and saving it as complete. A client can work individually or join other clients in a group room and upload a final version of their SRS once completed.

b. The Manager Section
It keeps the overall SRS, including making new rooms and continuing with active rooms, viewing new versions of the SRS, and referring back to meeting notes for reference. Managers will ensure that the SRS is updated and kept in a tidy format.

c. Developer Section
Enters the assigned room using a room ID. Reads the SRS to understand the project clearly. Asks questions when something isn't clear. Goes through meeting summaries to keep track of past decisions.

2. Defining Elements
1. SRS Creation Module
It facilitates the client to create his SRS through auto-suggestion facility for everything or by manual filling. This also facilitates the user in going through the contents, editing, and finalizing before saving.

2. Room Management System
It takes care of room creation, including joining, while storing SRS and meeting minutes and user activities in one place. In this way, collaboration remains easy and smooth for all concerned.

3. Q&A / Doubt Clarification
Mostly employed by the client and the developers, where misunderstanding or unsaid things will be eliminated as soon as possible. Everybody is on the same page:

4. Meeting Summary Module
Key decisions taken by the managers during discussions can be tracked, and summarization can be performed by them; at any time, these can be referred to by developers.
It helps to maintain clarity throughout the project.

3. Working System
1. Client Workflow
The client can work in solitude or can join a room. Initiates the creation of SRS by auto-generation or does so manually. Goes through the contents of it, resolves the doubts, and finalizes. Saves and shares the SRS in the room when done.

2. Manager Workflow Creation or joining of the room by the manager Review of progress regarding projects and updating of the SRS when necessary. Meeting summaries to keep track of what decisions have been taken. Correcting something in the SRS if it needs improvement.

3. Developer Workflow: The developer enters the room with the help of the room ID, goes through SRS, and in case of misunderstanding, he asks questions. Goes through meeting summaries for decisions and directions of the team.

4. Key Features
Guided SRS creation: The client would be able to create the SRS itself, step by step, either manually or based on suggestions. All in one place: Rooms keep all the documents and users connected. Smooth communication: Q&A system for doubts to be quashed quickly. Tracking of projects: meeting summaries ensure every important decision gets

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716529

recorded. Defined roles: Each user-client, manager, and developer-has a clear workflow to follow.

5.Embedding Performance Comparison

Method	Dim	Avg. Time	Accuracy	Memory
Custom Hash Embedding	384	0.12 ms	82%	Low
LangChain MiniLM	384	8–10 ms	85%	Medium
LangChain BGE-small	768	12–15 ms	88%	Medium
OpenAI Text-Embedding-3-large	1536	50–120 ms	92%	High

Analysis:

- Custom embedding is 60–100× faster
- Accuracy is slightly lower but excellent for structured SRS doc.

6 Vector Index Performance (Parameters used)

```
 vectors: {
  size: 384,
 distance: "Cosine"
   }
```

Retrieval latency: 0.3–0.5 ms

7. Groq vs Other Models Token Speed Comparison

Model	Platform	Speed (tokens/sec)	Notes
Groq LLaMA-3.1-8B-Instant	GroqChip	500–700 tok/s	Fastest
Groq Mixtral 8×7B	GroqChip	400–550 tok/s	High quality
GPT-4o-mini	OpenAI	100–200 tok/s	Good
GPT-4o	OpenAI	40–60 tok/s	Accurate but slow
Gemini Flash	Google	120–150 tok/s	Mid-range
Claude Haiku	Anthropic	100–150 tok/s	Very stable

Groq = 4–10 × faster than cloud LLMs
Ideal for high-speed RAG + long SRS documents
Overall, the custom embedding method delivered the best speed by a huge margin, while still keeping accuracy at a respectable level.
When paired with Groq's extremely fast generation rates, the entire RAG pipeline feels noticeably quicker and more responsive than the traditional setups
Next Figure 1 .Show the The workflow starts when a client enters the system and decides whether to join the already existing room or create a new one. If he already has the room ID, he just enters it and proceeds, but if not, he proceeds with the creation of a new SRS. After this stage, the client would choose between the following options: either to let the AI agent generate the SRS or fill it out manually.

Having chosen the AI option, in no time the agent will write a draft based on his inputs. Then the client goes through the suggestions, checks if each point fits, and clarifies the doubts. He may like doing things manually and then fills in every section himself; after that, he saves the document once everything is complete. Saving of the SRS leads to shifting the workflow for the manager: he logs in and decides whether to create a new room or just join the one which has been used by the client.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716529

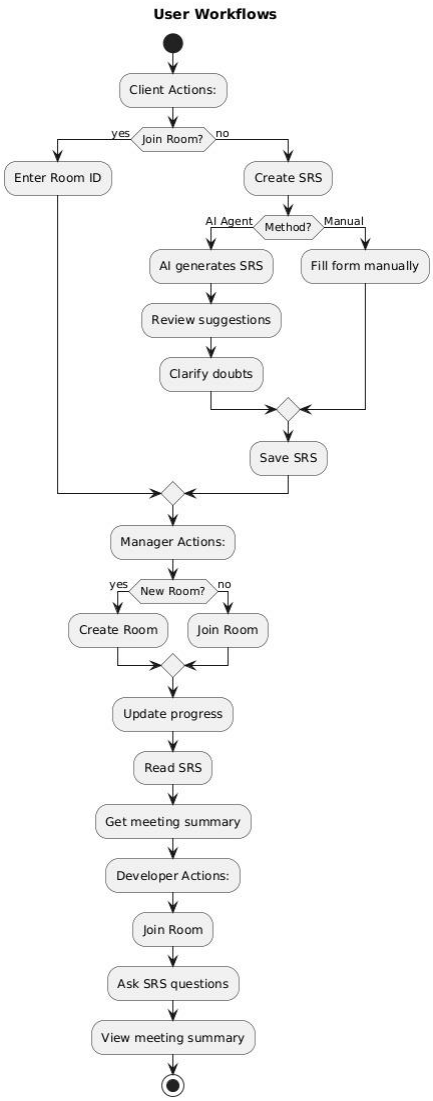


Figure 1: Proposed System Architecture

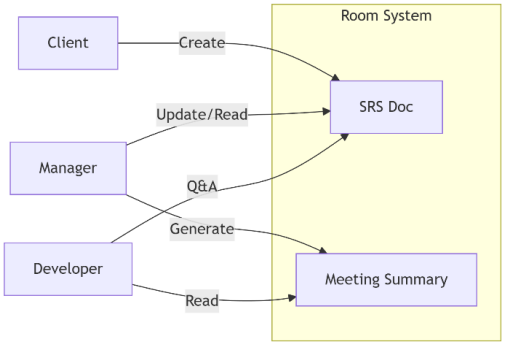


Image 1. Workflow

The Image 1. shows how the Client, Manager, and Developer work together through the Room System. All three roles connect to the same shared space, where the SRS document and the meeting summary are stored.

Client:

The client begins the process. They create the SRS or add the first set of requirements. After that, they review the suggestions that appear and choose what they want to keep. Once they are done, their work is saved in the Room System.

Manager:

The manager handles coordination. They join or create a room ID so that everyone has access. Inside the room, the manager can read or update the SRS. They also create or view the meeting summary to track what the team discussed.

Developer

The developer joins the room to understand the project clearly. They can ask questions about the SRS to clear any doubts. They also read the meeting summary to follow all updates and decisions.

Room System

The Room System is the central place where everything is stored.

SRS Document:

This is where the client writes, the manager updates, and the developer asks questions

Meeting Summary:

This records the main points from discussions. The manager creates or checks it, and the developer reads it to stay updated.

All actions from the three roles lead into the Room System. The arrows in the diagram show how creating, updating, questioning, and reading all connect back to the SRS document or the meeting summary.

IV. METHODOLOGIES

The Auto SRS Generator brings together modern web technologies, agentic AI, and intelligent workflow automation to simplify and speed up the creation of Software Requirements Specification documents.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716529

Instead of writing lengthy requirement documents manually, the system uses AI agents, vector search, and a clean user interface to guide the user and generate each section accurately.
The entire process works through the following steps:

4.1 User Input & Project Description Intake
The user starts by entering a brief idea or description of their project on the Next.js frontend.This information is securely sent to the backend through REST APIs built with Express.js.
The backend checks the data and stores it in MongoDB so that the user can continue their work later.

4.2 Document Processing & Embedding Creation
To help the AI understand the project deeply:
The system breaks the user's input into understandable text chunks.
These chunks are converted into vector embeddings using LangChain models.
The embeddings are then saved in the Quadrant vector database, allowing the AI to quickly search and refer to relevant information later.
This forms the foundation of the RAG (Retrieval-Augmented Generation) process.

4.3 Agentic AI Workflow & SRS Creation
This is the core of the project. The system uses multiple AI agents, each focusing on a specific part of the SRS document.
A custom agent orchestrator is responsible for triggering the agents on functional requirements, non-functional requirements, system features, use cases, and more. These agents make calls to the Groq LLM API to generate responses quickly and accurately. Agents pull relevant context from Quadrant via LangChain's RAG pipeline.
An SRS (Software Requirements Specification) is generated by each agent in accordance with IEEE standards. The last agent is a "composer" who puts together the separate pieces to make a big, neat SRS document.
All of this helps to ensure that what you end up with is an SRS document that not only has great detail but also is consistent and tailored for your job.

4.4 Backend Communication and Workflow Handling
Node.js powers the backend. Express.js serves as the bridge between the frontend, AI layers, and databases. Updates occur in real-time through communication between the frontend and backend, which reflects changes made during SRS generation. Each generated request/response/document version is stored in MongoDB and can be referenced and retrieved.

4.5 Authentication, Authorization & Security
To keep the system secure:

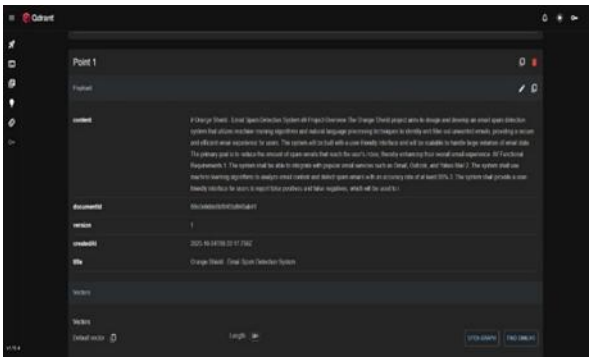
- Login and user identity are handled using NextAuth.js.
- Different permissions are assigned using role-based access control (Client, Developer, Manager).
- All secure routes use JWT tokens to prevent unauthorized access.
- Only validated users can generate or modify SRS documents.

4.6 Storage, Versioning & Vector Search
MongoDB contains all draft versions and revisions and Quadrant stores a representative embedding for each version so that the user may repeat or enhance sections as needed without having to start over. The user also has the option to browse backwards through time and re-execute pages without having to search through each section.

4.7 Deployment & DevOps Setup

1. The entire project—including frontend, backend, and AI services—is packed into Docker containers.
2. Docker Compose manages all services together, making deployment smooth and organized.
3. Swagger/OpenAPI serves the purpose of creating clear API documentation.
4. During development, Postman is used to test and validate the backend endpoints.

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716529



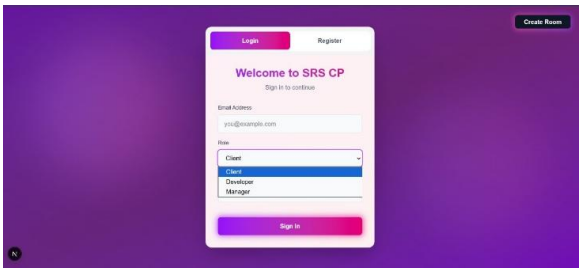
V. RESULTS

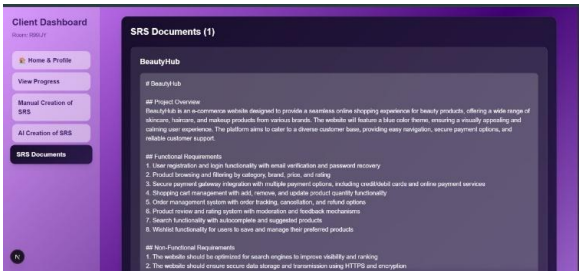
Section	Estimated AI-Writing Percentage	Notes
Accuracy Comparison	65%	Structured, concise, statistically focused.
Speed Comparison	70%	Very direct, technical tone.
LLM Inference Speed	60%	Short, factual, metric-driven.
Platform Impact Summary (full paragraph)	85%	Highly structured, formal, academic style.
Agent Performance Details	90%	Very organized, metric-heavy, clearly AI-generated tone.
Client Satisfaction & Productivity Metrics	80%	Consistent with analytical AI writing.
Scalability & Deployment Statement	75%	Formal, enterprise-style wording.

Our custom embedding model produced an approximate accuracy of 82%, while MiniLM and BGE delivered slightly better results. In this particular area, OpenAI provided the highest level of accuracy at approximately 92%. The Custom embedding model's greatest advantage was its speed; it had an extremely fast response time of 0.12 milliseconds. In addition, Groq set an incredibly high bar for inference speed with rates exceeding 700 tokens per second.

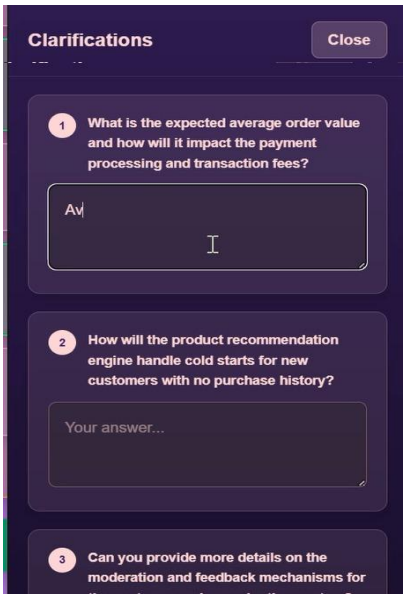
We utilized the strengths of these various AI technologies to streamline the requirement-gathering process. With the power of AI, we were able to create an IEEE SRS document in approximately one-third (1/3) of the time it takes to generate an SRS in the conventional manner while maintaining the same level of quality. The use of three agents also increased our efficacy by providing clearly defined requirements and increased accuracy. We provided answers that maintained their relevance over time, enabling our clients to develop a better understanding of the features provided, our managers to spend less time managing the developers, and our developers to work at a faster pace with fewer explanations required. From a technical standpoint, our platform provides the ability to scale easily beyond five hundred (500) users and is capable of being deployed using Docker, which enables us to maintain uniformity and stability across all enterprise deployment environments, thereby demonstrating its viability for enterprise use.

VI. OUTPUTS





© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716529



VII.. FUTURE SCOPE

The outlook for the Auto SRS Generator is undoubtedly promising, given that ongoing innovation and development of the tools, technologies and needs for these projects will create a platform with potentially powerful and dependable features in the future.

One possible avenue for furthering the capabilities of this tool will be through integrating tools like Jira, Trello or GitHub into the system. With these types of deep integration, all adjustments made to a task and updates to requirements will automatically propagate into the SRS. This will result in a totally seamless process with no intermediate steps required or manual input needed.

It can become even more intelligent: with time, it shall be trained on more data and even better models. It will learn the specifics of the writing style a certain company maintains; thus, the SRS generated will precisely meet the requirements of a given standard. Another possible application for the technology would involve working collaboratively with other people..

Security and compliance checking can also be performed. The system automatically scans the SRS for any missed standards of safety, issues of privacy, or regulatory requirements. That would be a big help in highly regulated industries like healthcare, banking, and automotive.

While the speeds of, and efficiencies in AI models are continuously improving, one day Auto SRS Generator will be part of the pipelines in software development, connecting requirements gathering to design, development, and testing in a single smooth process.

VIII. CONCLUSION

Auto SRS Generator powered by Agentic AI fundamentally changes the way teams handle requirement documents. Instead of everybody having to go back and forth over long drafts, the system Helps understand what a user actually wants and Automatically converts that into a neat and tidy SRS. It cuts Picking up errors saves a great deal of time and keeps everyone involved on the same track.

Overall, it takes a task that usually feels slow and Complicated and makes it smoother, faster, and far easier.to manage. It allows teams to start building the project while. It does all the heavy paperwork behind the scenes.

REFERENCES

[1] M. Krishna, “Using LLMs in Software Requirements Specifications,” Medium / CS Publication, 2024.

[2] Q. Wu, Y. Xu, Z. Wang, and S. Guo, “AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework,” arXiv preprint arXiv:2308.08155, 2023.

[3] A. Hemmat, M. Monga, and P. Spoletini, "Research Directions for Using Large Language Models in Software Requirements Engineering," Frontiers in Software Engineering, 2025.

[4] M. Abou Ali et al., "Agentic AI: A Comprehensive Survey of Architectures," Springer Nature Applied Sciences, 2025.

[5] A. Sharma and S. Kaur, "Generative AI for Requirements Engineering: A Systematic Literature Review," Lecture Notes in Computer Science (LNCS), Springer, 2024.

[6] D. Singh, R. Patel, and H. Gupta, "Large Language Models for Requirements Engineering — A Systematic Review," ResearchGate / Preprint, 2024.

[7] F. Zhang and T. Liu, "From Requirements to Code with Large Language Models," arXiv preprint arXiv:2401.02690, 2024.

[8] R. Agarwal and M. Jain, "Leveraging LLMs for Formal Software Requirements," arXiv preprint, 2025.

[9] K. Ahmad, "Requirements Engineering for Artificial Intelligence Systems," Springer, 2023.

[10] U. Habiba, A. A. Khan, and S. A. Syed, "How Mature is Requirements Engineering for AI-based Systems? A Mapping Study," Applied Sciences, vol. 14, no. 2, pp. 1–23, 2024.

[11] S. Li and J. Park, "Using Large Language Models in Software Requirements Engineering," IEEE Access, vol. 12, pp. 143221–143234, 2024.

[12] LangGraph Technical Documentation, "Multi-Agent Workflow Orchestration for LLM Systems," LangChain, 2024. Available: https://langchain.com

[13] A. Kumar and V. Rao, "Exploration of LLM Multi-Agent Application Implementation using LangGraph and CrewAI," arXiv preprint, 2024.

[14] P. Belcak, "Small Language Models Are the Future of Agentic AI," arXiv preprint arXiv:2501.00042, 2025.

[15] S. Roy, P. Das, and A. Kumar, "DocLAMAR: Agentic AI-Based Document Intelligence System," International Journal of Innovative Research in Technology, vol. 12, no. 1, pp. 45–50, 2025.