footnote151 footnote295 footnote31511

A Multi-Agent Requirement Engineering System - MARS

Project Team

Arrij Fawwad 22I-0755
Hamna Arshad 22I-1098
Zubair Khalid 22I-2475

Session 2022-2026

Supervised by

Dr. Arshad Islam

Co-Supervised by

Mr. Usama Imtiaz

Department of Computer Science

National University of Computer and Emerging Sciences
Islamabad, Pakistan

June, 2026

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Software development projects often face challenges not due to technical shortcomings, but because of poorly defined requirements that compromise the foundation of the system. Requirements engineering (RE) is widely recognized as one of the most critical yet error-prone phases of the development lifecycle. Conventional approaches rely heavily on manual effort and the availability of experts, making the process time-consuming, inconsistent, and prone to human error. This creates barriers for non-specialists, reduces efficiency, and increases the likelihood of costly mistakes. The Standish Group's CHAOS Report highlights this risk, showing that a clear statement of requirements contributes 13% to project success, while 12.3% of challenged projects are caused by incomplete or unclear specifications [4]. These figures underline how even small lapses in requirement clarity can trigger significant delays, rework, and in many cases, outright project failure. Traditional RE practices often produce Software Requirements Specification (SRS) documents that suffer from ambiguity, inconsistency, and conflicts, leading to miscommunication among stakeholders. Research by IBM underscores the importance of accuracy at the outset, showing that fixing a defect in production can cost up to 100 times more than correcting it during the requirements phase [2]. Ultimately, the recurring problems of incompleteness, ambiguity, and inconsistency in requirements undermine software quality and project success. These challenges highlight the urgent need for an intelligent, systematic solution that enhances requirement clarity, reduces errors, and streamlines the transition to development.

1.2 Motivation

The motivation for this project arises from the critical role that requirements engineering plays in determining software success and the persistent challenges associated with it.

Studies indicate that around 56% of software defects originate in the requirements phase [1], showing how errors at this stage create the most damaging ripple effects throughout development. These figures highlight the urgency of improving practices that ensure clarity, completeness, and consistency in requirements. At the same time, the requirements management market is projected to reach USD 3.2 billion by 2033 [3], with AI integration emerging as a key driver of growth. This trend reflects a broader industry recognition that traditional approaches are insufficient and that intelligent, automated support is increasingly essential. The combination of high technical risk and growing market demand provides strong motivation to explore solutions that enhance requirements engineering practices. Ultimately, the driving motivation behind this project is the opportunity to reduce costly downstream errors, improve project outcomes, and contribute to the evolving landscape of requirements engineering by addressing one of the most persistent pain points in software development.

1.3 Problem Solution

The proposed software, MARS (Multi-Agent Requirements Engineering System), is an AI-driven assistant designed to directly address the recurring challenges identified in the requirements engineering process. Traditional practices often result in incomplete, ambiguous, or conflicting specifications, which increase rework, miscommunication, and project failures. To overcome these limitations, MARS integrates automation, reasoning, and conversational interaction into a unified platform, ensuring requirements are captured, validated, and organized with greater accuracy and efficiency. The first objective of MARS is to engage users in a conversational elicitation process, enabling both technical and non-technical stakeholders to express requirements without the need for specialized expertise. Second, the system performs automated quality assurance, detecting ambiguity, conflicts, duplicates, and non-atomic requirements at the earliest stage, thereby reducing costly downstream errors. Third, it ensures proper categorization of requirements into functional and non-functional types, aligning them with established software engineering practices. Beyond validation and classification, MARS contributes to agile development workflows by generating user stories from refined requirements, making them directly actionable for development teams. Another key goal is to produce editable, standards-compliant Software Requirements Specification (SRS) documents, which include core sections such as Introduction, Scope, Functional and Non-Functional Requirements, and Glossary. At the same time, MARS allows the inclusion of elements, added and customized directly by users through the integrated chatbot, ensuring the document remains adaptable to project-specific needs. By meeting these objectives, MARS minimizes the reliance on manual effort, reduces the expertise barrier for high-quality requirements engineering, and accelerates the process of moving from stakeholder intentions to structured

specifications. The ultimate goal is to embed quality at the requirements stage, where errors are most costly, while making professional-grade requirements engineering more accessible, consistent, and adaptable across diverse project environments.

1.4 Stake Holders

1. Clients / End-Users – Provide the requirements and validate that the final SRS reflects their needs.
2. Business Analysts / Requirements Engineers – Use the system to elicit, refine, and structure requirements into a high-quality specification.
3. Project Managers – Depend on accurate, conflict-free requirements to plan resources, timelines, and deliverables.
4. Software Developers – Rely on the generated user stories and categorized requirements to guide implementation.
5. Quality Assurance (QA) Teams – Use the clarified requirements to design effective test cases and ensure product correctness.

Chapter 2

Project Description

2.1 Scope

The scope of this project is to design and implement an AI-assisted system that supports requirement elicitation, quality assessment, refinement, and partial automation of the Software Requirement Specification (SRS) document. The system will accept user input in text form or document upload, limited to English, and provide a chatbot-based interface to enable natural and guided elicitation of requirements. It will generate context-aware clarifying questions to resolve ambiguities and missing details through iterative interaction with users. Drafted requirements will be automatically analyzed for duplication, conflicts, and classification, with the system suggesting corrections where possible and escalating complex issues to users through human-in-the-loop support. The system will further automate the creation of a partially complete SRS, covering introduction, audience, scope, product perspective, functional and non-functional requirements, glossary, and user classes. Users will be able to select templates, refine sections, and add new requirements with chatbot assistance. Additionally, finalized requirements will be transformed into structured user stories to support agile development processes. However, the scope excludes functionalities such as software coding, testing, graphical modeling, or generating diagrams. It will not support multiple projects simultaneously or collaborative editing, and users will retain full responsibility for final review and validation. The system also does not address hardware or architectural constraints that may impact performance. By focusing specifically on elicitation, refinement, and structured documentation, the project establishes clear boundaries and ensures that users receive AI-driven support for requirement engineering without extending into development or operational activities.

2.2 Modules

The proposed system is divided into the following key modules, each addressing a specific phase of the requirement engineering process and collectively contributing to the

automated generation of high-quality software requirement specifications.

2.2.1 Conversational Requirement Elicitation & Drafting Module

This module enables users to interact with the web application through an AI-assisted conversational chatbot designed for requirement elicitation. The system engages users in iterative dialogue, asking clarifying questions to resolve ambiguities and ensure completeness. User-provided inputs are then organized into a preliminary structured format that serves as the foundation for subsequent refinement and validation.

1. Provides an interactive conversational assistant to guide the elicitation process.
2. Generates clarifying questions to capture precise and complete requirements.
3. Accepts both free text inputs and structured lists of requirements.
4. Produces an initial structured draft of requirements for further refinement.

2.2.2 Requirement Quality Assurance Module

This module ensures the clarity, consistency, and accuracy of drafted requirements by applying automated quality checks. The system identifies issues such as ambiguity, lack of atomicity, duplication, and logical conflicts, and suggests corrective actions to improve the overall reliability of the requirements.

1. Enforces atomicity by breaking down complex requirements into simpler, precise statements.
2. Detects duplicate and conflicting requirements for resolution.
3. Highlights conflicting or incomplete requirements for user clarification.

2.2.3 Requirement Refinement & Categorization Module

This module further enhances validated requirements by refining their structure and categorizing them into appropriate groups. Functional and non-functional requirements are distinctly identified to support systematic documentation and traceability. Any conflicts that cannot be automatically resolved are logged for later human verification.

1. Automatically classifies requirements into functional and non-functional categories.

2. Provides a categorized view of requirements for improved organization and accessibility.

3. Refines requirement phrasing and maintains a record of unresolved conflicts for human review.

2.2.4 User Story Generation Module

This module bridges the gap between requirements engineering and agile development by converting validated requirements into structured user stories. Each user story is enriched with essential elements such as a title, description, and acceptance criteria, ensuring clarity and alignment with agile practices.

1. Generates user stories with structured elements (title, description, acceptance criteria).

2. Cost and risk analysis of user stories using the project input elicited during the elicitation phase.

3. Prioritization of user stories to aid in sprint planning and resource allocation.

2.2.5 SRS Document Generation & Editing Module

This module automates the creation of a Software Requirement Specification (SRS) document, integrating all validated requirements, user stories, and glossary terms into a coherent format. The system adheres to IEEE standards while also supporting user-provided templates for customization. Users can refine the document interactively with chatbot assistance and export it in multiple formats for practical use.

1. Assembles an editable SRS document including introduction, scope, functional and non-functional requirements, and glossary.

2. Adheres to IEEE standards while supporting customizable templates provided by the user.

3. Offers real-time chatbot-assisted editing and refinement of the document.

4. Exports the finalized SRS into multiple formats such as PDF, DOCX, and Markdown.

2.3 Tools and Technologies

Frontend (Web Application Interface)

- React.js
- HTML5, CSS3, JavaScript
- Bootstrap

Backend (Application Logic and APIs)

- Python (Flask/FastAPI/Django)
- Node.js (optional)

AI/ML and NLP Technologies

- NLTK
- PyTorch

Database and Storage

- PostgreSQL / MySQL
- MongoDB (optional)
- File Storage

Document Processing & Export

- python-docx
- Pandoc

2.4 Work Division

For each module and respective Feature, assign responsibility to a team member

Table 2.1: Table 1

Name	Registration	Responsibility / Module / Featureure
Mr. Zubair	22i-2475	Module 1- Feature 1, Module 2- Feature 1, Module 3- Feature 1, Module 4- Feature 2, Module 5- Feature 3
Miss. Arrij	22i-0755	Module 1- Feature 2-3, Module 2- Feature 1, Module 3- Feature 2, Module 4- Feature 1, Module 5- Feature 4
Miss. Hamna	22i-1098	Module 1- Feature 4, Module 2- Feature 3, Module 3- Feature 3, Module 4- Feature 3, Module 5- Feature 2
Team	-	Module 1- Feature 1, Module 2- Feature 2, Module 5- Feature 1

2.5 TimeLine

Table 2.2: Project Iteration Plan

Iteration #	Time Frame	Tasks / Modules
01	Sept–Oct	Development of Conflict and Duplicate Detection Model, including data preprocessing and validation.
02	Nov–Dec	Implementation of Ambiguity Detection, Requirement Classification, and Refinement mechanisms.
03	Jan–Feb	Conversational Requirement Elicitation and Automated User Story Generation.
04	Mar–Apr	Integration of SRS Editor, Chatbot Interface, Requirement Filtering, and Preliminary Cost–Risk Analysis.

Chapter 3

Conclusions and Future Work

3.1 Conclusion

The proposed project outlines the design and implementation of an AI-assisted system that enhances the requirements engineering process by focusing on elicitation, refinement, quality assurance, and structured documentation. Through its modular architecture, the system provides an interactive chatbot interface for iterative requirement gathering, applies automated checks for consistency and accuracy, and transforms validated requirements into both structured user stories and a partially automated SRS document aligned with IEEE standards. The integration of natural language processing techniques with document generation tools ensures that users benefit from AI-driven support while maintaining human oversight for final validation. By clearly defining the system's scope, excluding aspects such as coding, testing, or architectural design, the project ensures a targeted approach that addresses some of the most pressing challenges in requirement engineering. The systematic work division and structured timeline further highlight the feasibility and collaborative nature of the development process. Overall, this project contributes toward bridging the gap between requirement elicitation and agile development practices, offering a practical, intelligent, and user-friendly solution that streamlines the creation of reliable and high-quality software requirement specifications.

3.2 Future Work

While the proposed system provides a structured and AI-assisted approach to requirement elicitation and documentation, there remain several opportunities for further enhancement. A key direction for future work is the development of a multi-perspective requirement elicitation engine capable of capturing requirements from diverse stakeholder viewpoints, ensuring broader coverage and reducing bias in the gathered specifications.

Another avenue involves incorporating adaptive learning mechanisms, such as Cross Talk Instance Memory, to enable the system to learn from previous interactions and refine its responses over time, thereby improving accuracy and contextual relevance. In addition, expanding interoperability with widely used software project management tools, such as GitHub, spreadsheets, and Trello, can significantly improve the system's practical utility by enabling seamless integration into existing workflows. These extensions would not only broaden the system's applicability but also strengthen its role in supporting collaborative, agile, and large-scale software development projects.

Bibliography

[1] Shankar Acharya. 7 principles of software testing, 2020. https://medium.com/@er.shankar.acharya/7-principles-of-software-testing-2d1ce74a Accessed: 2025-09-08.

[2] Functionize. The cost of finding bugs later in the sdlc, 2023. https://www.functionize.com/blog/the-cost-of-finding-bugs-later-in-the-sdlc, Accessed: 2025-09-08.

[3] Market Research Intellect. Global requirements management tools market size forecast, 2023. https://www.marketresearchintellect.com/product/global-requirements-management-t Accessed: 2025-09-08.

[4] Standish Group. The chaos report, 1995. https://www.csus.edu/indiv/v/velianitis/161/chaosreport.pdf, Accessed: 2025-09-08.