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 Introduction to the Domain

Requirements Engineering (RE) is a fundamental area of software engineering that focuses on identifying, analyzing, documenting, and managing the requirements of a system. By creating a common understanding between developers and stakeholders, it guarantees that the finished product will meet both organisational goals and user demands. By providing a structured process for capturing both functional and non-functional requirements, RE plays a critical role in improving communication, guiding design and implementation, and supporting successful software delivery.

1.2 Problem Statement

Software development projects frequently fail because of poorly defined requirements that jeopardise the system's foundation rather than technical flaws. Despite its importance, requirements engineering (RE) is still one of the stages of the development lifecycle that is most prone to mistakes. Traditional methods are time-consuming, inconsistent, and prone to human error because they mainly rely on manual labour and the availability of experts. This dependence hinders non-specialists, lowers productivity, and raises the risk of expensive errors.

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 stem from incomplete or unclear specifications [6]. Research by IBM further underscores that fixing a defect in production can cost up to 100 times more than correcting it during the requirements phase [2]. These figures demonstrate how even small lapses in requirement clarity can trigger significant delays, rework, or outright project failure.

Among the most critical challenges in RE are the issues of conflict and duplicate

requirements. Conflicting requirements often arise when multiple stakeholders provide overlapping or contradictory inputs, while duplicates introduce redundancy that obscures traceability and inflates project complexity. Both challenges increase cognitive load for analysts, prolong validation cycles, and elevate the risk of inconsistency in the final specification.

This project specifically focuses on developing an intelligent agent-based system that improves conflict and duplicate detection within requirements engineering. By systematically identifying and resolving these issues, the proposed solution aims to enhance requirement clarity, reduce errors, and streamline the transition to development.

1.3 Motivation

The importance of requirements engineering (RE) in determining the success of software development serves as the driving force behind this project. The efficiency and dependability of the entire development process are directly improved by better requirements management, refinement, and capture since they specify what is to be built. Teams can decrease rework, shorten development cycles, and enhance stakeholder collaboration by making sure that requirements are explicit, consistent, and well-structured from the beginning.

Studies conducted by the industry highlight how urgent this problem is. The importance of improving requirement clarity and validation is highlighted by the fact that 56% of all system defects are introduced during the requirements phase [1]. Furthermore, it is anticipated that the requirements management tool market will reach 3.2 billion USD by 2033, with AI integration serving as a key catalyst for this growth [5]. This pattern demonstrates the economic importance of RE as well as the rising demand for automated, intelligent solutions.

This project seeks to leverage intelligent support to make requirements engineering more accessible and systematic, lowering the reliance on extensive manual effort and specialized expertise. In doing so, it aims to enable software teams to focus more on building solutions rather than resolving misunderstandings or correcting errors later in the lifecycle. With the software industry increasingly embracing automation and AI-driven tools, there is strong motivation to explore how these technologies can optimize RE practices and bring measurable improvements to project outcomes.

The motivation behind this project is to make requirements engineering smarter, faster, and less error-prone, thereby strengthening the entire software development process. By embedding quality at the requirements stage, the project aims to improve efficiency, reduce risks, and deliver more dependable outcomes across diverse software projects.

1.4 Existing Solutions

Several AI-assisted tools and platforms have been introduced to support requirements engineering and related activities. These systems aim to automate aspects of requirements authoring and management, but their focus is often limited to specific functions rather than providing a comprehensive, quality-driven approach. Table 1.1 provides an overview of notable solutions currently in the market.

Although these systems demonstrate the potential of AI to support requirements-related tasks, they reveal notable shortcomings. **Current tools lack intelligent requirement suggestions, rigorous quality-checking, conflict detection, and version control for evolving documentation.** These gaps highlight the need for a more comprehensive solution that combines elicitation, validation, refinement, and documentation within a single intelligent framework.

Table 1.1: Comparison of existing AI-assisted requirements tools and their limitations

System	Key Features	Identified Gaps
ScopeMaster	Checks grammar, clarity, complexity, and testability of requirements.	Does not provide requirement suggestions; lacks cross-checking for duplicates/conflicts.
Azure Copilot4DevOps	Generates user stories, tasks, and test cases to support DevOps workflows.	Focused on task generation rather than quality assurance; no rigorous requirement validation. For Azure workflows only.
WriteMyPRD	Quickly generates Software Requirements Specification (SRS) documents with minimal input.	Prioritizes speed over quality; lacks ambiguity/conflict checks; minimal elicitation assistance.
Beam's PRD AI	Produces fast and structured Product Requirement Documents (PRDs).	Limited to document generation; no requirement suggestions or conflict detection.

1.5 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 key objectives and features of MARS are outlined as follows:

1. **Conversational Requirement Elicitation:** Engages users in a chatbot-based elicitation process, enabling both technical and non-technical stakeholders to express requirements naturally, without requiring specialized expertise.

2. **Automated Requirement Quality Assurance:** Performs automated checks for conflicts, duplicates, and non-atomic requirements at the earliest stage, thereby reducing costly downstream errors.

3. **Requirement Classification and Refinement:** Ensures categorization of requirements into functional and non-functional types, aligning them with established software engineering practices, while refining their phrasing and clarity.

4. **User Story Generation for Agile Workflows:** Converts refined requirements into actionable user stories enriched with titles, descriptions, and acceptance criteria, making them directly usable for development teams.

5. **SRS Document Generation and Editing:** Produces editable, Software Requirements Specification (SRS) documents, covering sections such as Introduction, Scope, Functional and Non-Functional Requirements. The system also supports chatbot-assisted customization and integration of project-specific details.

6. **Accessibility and Adaptability:** Minimizes reliance on manual effort, lowers the expertise barrier for high-quality requirements engineering, and ensures that documents remain adaptable to diverse project environments.

By meeting these objectives, MARS embeds quality in the requirements stage, where errors are most costly, while making professional-grade requirements engineering more accessible, consistent, and efficient.

1.6 StakeHolders

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 Requirements Engineering System** that facilitates requirement elicitation, quality assessment, refinement, and assisted creation of the Software Requirements Specification (SRS) document. The system is limited to handling inputs in English and will primarily operate through a **chatbot-based interface** to enable interactive and guided elicitation of requirements.

2.1.1 Functional Scope

2.1.1.1 Requirement Elicitation

This component employs the **RASA NLP pipeline** to capture natural language inputs through conversational interaction. [4] The chatbot guides users by asking context-aware clarifying questions, thereby resolving ambiguities and addressing missing details.

2.1.1.2 Structured Requirement Generation

The system applies **SpaCy** and **regular expressions (Regex)** to transform elicited requirements into a structured format. This process enables classification into functional and non-functional categories and ensures consistency by normalizing requirements for subsequent analysis.

2.1.1.3 Conflict and Duplicate Detection

To ensure quality and coherence, requirements are represented as embeddings using the **Sentence Transformer (all-MiniLM)** model. Semantic similarity is identified through **K-Means clustering**, while potential conflicts and duplicates are detected using a hybrid approach that combines **Graph Neural Networks (GNNs)** for exploratory detection and **Siamese Networks** for supervised classification. Software requirements Dataset for Conflict Detection will be used for training the models. [3]

2.1.1.4 Requirement Refinement

Refinement is achieved through the **Hugging Face Transformers library**, using models such as **GPT Neo** and **Llama**. The system assists users by rephrasing vague requirements, resolving conflicts, and suggesting clearer alternatives, thereby enhancing the overall quality of the requirement set.

2.1.1.5 User Stories and System Features

Elicited requirements are transformed into structured user stories using **SpaCy**,**Regex**. These user stories follow agile development conventions, thereby supporting downstream software design and development activities.

2.1.1.6 SRS Document Generation and Export Module

The system provides an **editor-assisted environment** for creating the Software Requirements Specification. It generates an **initial draft** that includes standard sections such as the introduction, scope, product perspective, and both functional and non-functional requirements. Users can refine, expand, and customize this draft interactively with chatbot support, and the finalized SRS can be exported in multiple formats, such as DOCX and PDF. This functionality is implemented using a hybrid approach that combines **rule-based templates**, **GPT Neo**, and **Python-based export modules**. For the editor interface, **Streamlit** will be employed to provide an intuitive, interactive environment for editing and refining requirement documents.

2.1.2 Exclusions

The system does not provide functionality for software coding, testing, or deployment. It excludes graphical modeling and automatic generation of UML or ER diagrams, does not support multi-project management or collaborative editing, and does not account for hardware or architectural design constraints.

2.1.3 Boundaries and Impact

By focusing on **elicitation, conflict detection, refinement, and structured documentation**, the project establishes clear boundaries within requirements engineering. It reduces manual effort, improves requirement quality, and provides automated support in SRS generation while leaving final validation and approval under the responsibility of human stakeholders.

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 System Architecture

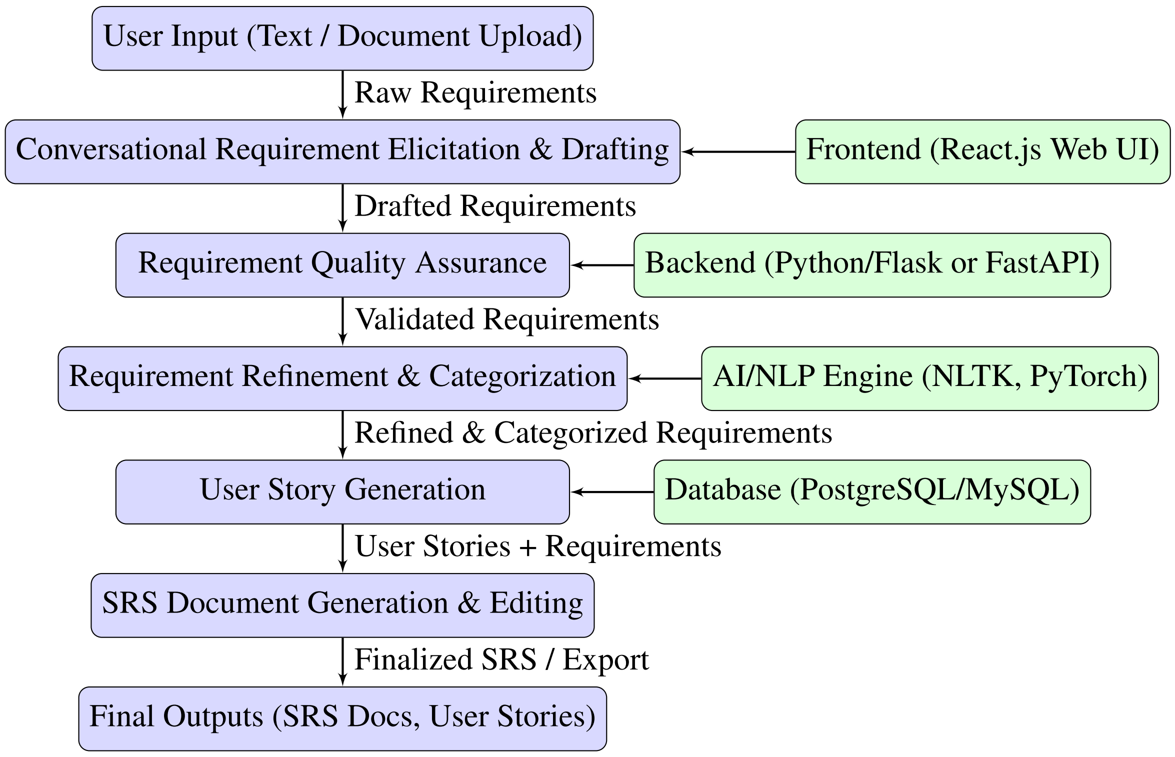
User Input (Text / Document Upload)
Raw Requirements
Conversational Requirement Elicitation & Drafting
Frontend (React.js Web UI)
Drafted Requirements
Requirement Quality Assurance
Backend (Python/Flask or FastAPI)
Validated Requirements
Requirement Refinement & Categorization
AI/NLP Engine (NLTK, PyTorch)
Refined & Categorized Requirements
User Story Generation
Database (PostgreSQL/MySQL)
User Stories + Requirements
SRS Document Generation & Editing
Finalized SRS / Export
Final Outputs (SRS Docs, User Stories)

Figure 2.1: Pipe-and-Filter System Architecture of MARS

The architecture of the proposed system follows a **pipe-and-filter model**, where each stage of the requirement engineering process is represented as a modular component connected in sequence. User input flows through a series of processing stages, with the output of one stage serving as the input for the next. This ensures that requirements are progressively refined, validated, and transformed into structured artifacts such as user stories and SRS documents.

At the entry point, users interact with the system through a **frontend web application**, implemented in React.js, which provides both text-based input and document upload options. The frontend connects to the **backend services**, developed in Python, responsible for orchestrating workflows and exposing APIs. Core intelligence is handled by the **AI/NLP engine**, which leverages libraries such as NLTK and PyTorch to perform requirement elicitation, ambiguity detection, conflict resolution, and classification.

Validated requirements and generated artifacts are stored in the **database layer** (PostgreSQL/MySQL or MongoDB for unstructured storage), ensuring persistence and traceability. The pipeline includes modules for conversational elicitation, quality assurance, refinement, user story generation, and SRS assembly, each functioning as independent but interconnected processing blocks. Additionally, the system integrates a **document processing component**, using tools like `python-docx` and Pandoc, to support export into multiple formats such as PDF, DOCX, and Markdown.

This modular architecture promotes scalability, maintainability, and flexibility. For example, improvements to the AI models or integration of new document templates can be performed without affecting the entire system. By adopting the pipe-and-filter design, the architecture ensures that requirements flow in a structured manner from raw input to finalized documentation while maintaining opportunities for human-in-the-loop interaction and feedback.

2.4 Tools and Technologies

Frontend (Web Application Interface)

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

Backend (Application Logic and APIs)

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

AI/ML and NLP Technologies

- NLTK
- SpaCy
- RASA
- Hugging Face Transformers
- Sentence Transformers
- Regex
- PyTorch

Database and Storage

- MySQL
- File Storage

Document Processing & Export

- python-docx
- WYSIWYG Editor
- Pandoc

2.5 Work Division

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

Table 2.1: Work Division

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

2.6 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.

Bibliography

[1] Mustapha Ahmad and Zainab Khan. A quantitative study to identify critical requirement engineering challenges in the context of small and medium software enterprises. *ResearchGate*, 2017.

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

[3] Garima2751. Tf_cdn dataset repository. https://github.com/garima2751/TF_CDN/tree/main/Da
2022.

[4] RasaHQ. Rasa: Open source conversational ai. https://github.com/RasaHQ/rasa, 2025.

[5] Verified Market Reports. Requirements management tools market report. https://www.verifiedmarketreports.com/product/requirements-management-tools-mark
2024.

[6] Standish Group. The chaos report. https://www.csus.edu/indiv/v/velianitis/161/chaosrepo
1995.