Project Synopsis

For

An AI-Driven System for Automated Requirements Engineering from Unstructured Stakeholder Communication

07/02/2026

Prepared by: -

Specialization	SAP ID	Name
DevOps	500122740	Shreyash Shivhare
DevOps	500123425	Sreyas Sharma
DevOps	500123468	Devanshi Jain
DevOps	500124202	Sreyas Sharma

Mentor: -

Dr. Mitali Chugh

[Associate Professor and Program Lead DevOps]

School Of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007. Uttarakhand

Table of Contents

1	INTRODUCTION	
	1.1 Purpose of the Project	The purpose of this project is to develop an **AI-Driven Automated Requirements Engineering System** that automatically extracts, classifies, clusters, prioritizes, and summarizes software requirements from natural language inputs such as stakeholder interviews, emails, and requirement documents. Traditional requirements engineering processes are manual, time-consuming, and prone to ambiguity and errors. This system aims to reduce human effort, improve requirement accuracy, and accelerate the software development lifecycle
	1.2 Target Beneficiary	• Software development teams • Requirement analysts and system analysts • Project managers • Software organizations and startups • Academic researchers working in Requirements Engineering and NLP
	1.3 Project Scope	The proposed system operates in the software development and requirements engineering domain. It processes natural language requirement documents and generates structured outputs such as classified requirements (functional/non-functional), clustered requirement groups, prioritized lists, and summarized reports. **Objectives:** • Automate requirement extraction from text • Classify requirements using machine learning • Cluster similar requirements for organization • Prioritize requirements based on importance • Generate structured summarized reports **Deliverables:** • NLP-based requirement processing model • Basic web-based prototype interface • Preprocessed datasets and trained models • Final project report and demonstration
	1.4 References	[1] K. Lam, M. H. Bhuiyan, M. S. Islam, A. H. Chowdhury, Z. A. Bhuiyan, and S. Ahmmed, “Co-pilot for project managers: Developing a PDF-driven AI chatbot for facilitating project management,” IEEE Access, vol. 13, pp. 43079–43096, 2025, doi: 10.1109/ACCESS.2025.3548519. [2] V. Siddeshwar, S. Alwidian, and M. Makrehchi, “A systematic review of AI-enabled frameworks in requirements elicitation,” IEEE Access, vol. 12, pp. 154310–154328, 2024, doi: 10.1109/ACCESS.2024.3475293.

		[3] M. A. Umar and K. Lano, "Advances in automated support for requirements engineering: A systematic literature review," Requirements Engineering, vol. 29, pp. 177–207, 2024, doi: 10.1007/s00766-023-00411-0. [4] G. Voria, F. Casillo, C. Gravino, G. Catolino, and F. Palomba, "RECOVER: Toward requirements generation from stakeholders' conversations," IEEE Transactions on Software Engineering, vol. 51, no. 6, pp. 1912–1927, 2025, doi: 10.1109/TSE.2025.3572056.
2	PROJECT DESCRIPTION	
	2.1 Reference Algorithm	The system follows an **NLP-based requirements processing pipeline** consisting of the following steps: 1. Input requirement text collection 2. Text preprocessing (tokenization, stop-word removal, stemming/lemmatization) 3. Requirement extraction using NLP techniques 4. Requirement classification using Machine Learning models 5. Requirement clustering using similarity-based clustering algorithms 6. Requirement prioritization using scoring techniques 7. Requirement summarization and structured output generation **Required Data Structures:** • Text datasets stored as structured CSV/JSON • Vectorized representations using TF-IDF matrices • Requirement clusters stored using array/list structures
	2.2 Characteristic of Data	The dataset used in the project consists of **natural language requirement documents**, including software requirement specifications, stakeholder requirement statements, and publicly available requirement datasets. **Primary Data Sources:** • Public requirements datasets (PURE dataset, Kaggle SRS datasets) • Sample requirement documents prepared for experimentation **Secondary Data Sources:** • Research paper datasets • Open-source software requirement repositories **Sampling Technique:** Random sampling of requirement sentences from collected datasets. **Statistical / Processing Methods:** • Text preprocessing using NLP techniques • Feature extraction using TF-IDF • Machine learning classification models

		• Clustering algorithms for grouping similar requirements
	2.3 SWOT Analysis	**Strengths** • Automates manual requirements engineering tasks using AI and NLP • Reduces time, human effort, and requirement ambiguity • Improves requirement organization and documentation quality • **Weaknesses** • Accuracy depends on dataset quality and model training • Requires preprocessing and computational resources **Opportunities** • Can be integrated into software development lifecycle tools • Useful for startups and organizations managing large requirement datasets **Threats** • Rapid changes in NLP technologies • Availability of commercial requirement automation tools
	2.4 Project Features	• Upload or input natural language requirement documents • Automatic requirement extraction and preprocessing • Classification of requirements (functional/non-functional) • Clustering of similar requirements • Requirement prioritization based on importance • Generation of summarized structured requirement reports • Basic web interface for user interaction
	2.5 User Classes and Characteristics	**Requirement Analysts:** Use the system to process requirement documents and generate structured outputs. **Software Developers:** Use structured requirement reports for system development planning. **Project Managers:** Use prioritized requirement lists for project planning and decision-making. **Researchers/Students:** Use the system for experimentation and academic research.

	2.6 Design and Implementation Constraints	• The system will run on standard computing devices with moderate memory requirements. • Implementation will use Python, NLP libraries (NLTK/spaCy), and machine learning frameworks. • A web-based interface will be developed for accessibility. • The system requires internet access for deployment and dataset retrieval (if applicable). • Security considerations include safe handling of uploaded requirement documents and restricted user access.
	2.7 Design diagrams	The following design diagrams will be prepared for the system: • Use Case Diagram showing user interactions with the requirement processing system • Class Diagram representing system classes such as Requirement, User, Processing Module, and Report Generator • Activity Diagram describing the workflow of requirement processing • Sequence Diagram illustrating interaction between system modules • Data Flow Diagram (DFD) representing input requirement flow and processed outputs • State Diagram representing requirement processing states
	2.8 Assumption and Dependencies	**Assumptions** • Users will provide requirement documents in readable text format. • Sufficient dataset availability for training and testing machine learning models. • Users will have internet-enabled systems to access the web interface. **Dependencies** • NLP libraries such as NLTK, spaCy, or Scikit-learn • Availability of training datasets for requirement classification • Hosting environment for deployment of the web interface
3	SYSTEM REQUIREMENTS	
	3.1 User Interface	A web-based user interface will allow users to: • Upload requirement documents • View processed classified and clustered requirements • Access summarized requirement reports • Download structured outputs
	3.2 Software Interface	The system modules include: • Input module for requirement upload • NLP processing module for text preprocessing • Machine learning module for classification and clustering

		• Reporting module for generating summarized outputs Communication between modules will occur through internal APIs implemented using Python-based frameworks.
	3.3 Database Interface	The system will use a lightweight database (SQLite / MongoDB) to store: • Uploaded requirement documents • Processed requirement outputs • Model results and logs
	3.4 Protocols	• HTTP/HTTPS protocol for web communication • Secure file upload handling • Standard REST-based communication between frontend and backend modules
4	NON-FUNCTIONAL REQUIREMENTS	
	4.1 Performance requirements	• The system should process uploaded requirement documents within a reasonable processing time depending on document size. • The application should support multiple document uploads without system failure. • Response time for report generation should remain efficient under moderate usage conditions.
	4.2 Security requirements	• User-uploaded requirement documents must be securely stored and accessed only by authorized users. • Secure communication should be maintained using HTTPS. • Access control mechanisms should restrict unauthorized system usage.
	4.3 Software Quality Attributes	**Adaptability:** The system can be updated to support new datasets, algorithms, or requirement formats with minimal modification. **Availability:** The system will be available through a web-based interface accessible to authorized users. **Correctness:** Machine learning and NLP models will ensure accurate extraction, classification, and summarization of requirements. **Flexibility:**

		The system allows integration of additional models and features without major structural changes. **Interoperability:** The system can interact with external software tools and datasets through standard APIs and file formats. **Maintainability:** Modular architecture enables easy debugging, updating, and system maintenance. **Portability:** The application can run on multiple operating systems supporting Python environments. **Reliability:** The system ensures consistent processing results under normal operating conditions. **Reusability:** Developed modules such as preprocessing, classification, and clustering can be reused in other NLP applications. **Robustness:** The system can handle incomplete or noisy requirement text inputs effectively. **Testability:** Individual modules can be tested independently using predefined datasets. **Usability:** A simple web interface enables easy interaction for both technical and non-technical users.
5	Other Requirements	• The system should support standard text file formats (PDF, TXT, DOCX). • The application should allow exporting processed requirement reports. • The system should support future scalability for handling larger datasets.
Appendix A: Glossary		• **NLP:** Natural Language Processing • **ML:** Machine Learning • **SRS:** Software Requirement Specification • **Requirement Classification:** Categorizing requirements into functional and non-functional types

	• **Requirement Clustering:** Grouping similar requirements based on similarity • **Requirement Prioritization:** Ranking requirements based on importance or impact
Appendix B: Analysis Model	The project uses an **NLP-based requirement processing model** consisting of text preprocessing, feature extraction, machine learning classification, clustering, prioritization, and summarization modules forming the complete requirement processing pipeline.
Appendix C: Issues List	• Dataset availability and quality may affect model accuracy • Requirement ambiguity may reduce classification precision • Model performance may require periodic retraining with updated datasets