**Project Synopsis:** **FF No 180**

**Project Title: Automated Requirement Writing**

**Introduction:**
The Automated Requirement Writing project leverages Artificial Intelligence (AI) and Natural Language Processing (NLP) to streamline the process of extracting, classifying, and documenting software requirements. Traditionally, requirement gathering is time-consuming, error-prone, and inconsistent, often involving manual review of documents such as Word, PDF, Excel, and emails. Our system automates this process by analyzing multiple input formats, classifying functional and non-functional requirements, and generating IEEE-compliant requirement documents. It also integrates with collaboration tools such as JIRA and Confluence for task tracking and version control, ensuring efficiency, traceability, and reduced manual effort.

**Literature Survey:**

**Table 1. Literature Survey of existing literature**

| Sr. No | Paper Title | Objectives claimed by authors in their paper | Methodologies used | Outcomes (from result & Discussion) | Gap Identified (Generally from Conclusion & Future scope) |
|---|---|---|---|---|---|
| 1. | An Approach towards Automation of Requirements Analysis | Automate object-oriented analysis from natural language requirements (identify actors, use cases, classes, relations). | NLP-driven CASE tool (R-TOOL); object-oriented mapping; pronoun/ambiguity resolution; ATM case study. | Successfully generated class diagrams from English text; resolved some ambiguities. | Struggles with incomplete/ambiguous inputs; needs stakeholder clarifications and broader domain coverage. |
| 2. | Advances in Automated Support for Requirements Engineering: A Systematic Literature Review | Synthesize advances in automation across RE phases (elicitation, analysis, validation, management). | Systematic review of tools/techniques (NLP, ML, formal methods) and their evaluation contexts. | Automation reduces effort, improves consistency, detects ambiguities early. | Limited end-to-end automation; weak integration with project workflows; difficulty with unstructured/multi-format inputs. |
| 3. | Automated Validation of Requirement Reviews: A Machine | Automate validation of review comments to improve requirement review efficiency. | ML classifiers (Random Forest, SVM, Naive Bayes) trained on review- | Significantly filtered invalid/irrelevant comments, speeding up reviews. | Performance depends on labeled data quality/size; may misclassify context-heavy comments. |

| | | | | | |
|---|---|---|---|---|---|
| | Learning Approach | | comment datasets. | | |
| 4. | Automated Requirement Sentences Extraction from SRS | Separate requirement sentences from background text in SRS documents. | NLP with rule-based sentence filtering + classification on real SRS. | Improved precision/recall for requirement-sentence detection. | Struggles with complex syntax, implicit requirements, and domain-specific vocabulary. |
| 5. | How Automate Requirements Engineering System Effects and Support Requirement Engineering | Evaluate how automated RE systems affect quality, speed, and traceability. | Case studies of projects using AI/NLP/CASE-integrated tooling. | Better requirement quality, fewer errors, faster documentation; improved traceability. | Limited adaptability across diverse domains; reliance on tool-specific formats; handling highly unstructured data remains hard. |
| 6. | An Error-Analysis Study from an EFL Writing Context: Human and Automated Essay Scoring Approaches | Compare AES systems with human scoring in EFL contexts; analyze error patterns. | Automated Essay Scoring (NLP + statistical features) vs. human raters on EFL essays. | AES is faster/more consistent; discrepancies on nuanced language/creativity/context. | Automated systems misread idioms/cultural cues; human oversight needed for complex language. |
| 7. | Automated Requirements Extraction for Scientific Software | Extract software requirements from scientific literature/documentation automatically. | NLP with domain dictionaries & pattern matching applied to scientific texts. | Efficient extraction using tailored term sets/patterns in scientific domains. | Limited portability; requires heavy domain customization and maintenance of dictionaries. |
| 8. | Automated Analysis of Requirement Specifications | Provide measurable quality indicators for NL requirements to aid managers/assurance. | ARM tool: text analysis of imperatives, weak phrases, readability, structure; applied across NASA specs. | Found ambiguity, weak structures, organization issues; gave actionable quality metrics. | Lacks semantic understanding/correctness checks; difficulty distinguishing prescriptive vs. descriptive statements. |
| 9. | Automated Requirements Engineering Framework for Agile Model-Driven Development | Bridge requirements to models in agile via automation for better alignment/traceability. | MDD integrated with agile; automatic transformation to PIM/PSM; | Reduced manual modeling effort; better requirements–design alignment; | Needs consistent, well-defined inputs; limited for highly volatile/ambiguous requirements. |

| | | | | | |
|---|---|---|---|---|---|
| | | | project applications. | enhanced traceability. | |
| 10. | Automated Requirement Contradiction Detection through Formal Logic and LLMs | Detect contradictions in requirements more robustly than logic-only systems. | Hybrid of formal logic reasoning and LLMs (e.g., GPT) for contextual checks. | Higher contradiction-detection rates vs. logic-only baselines; better context handling. | Sensitive to prompt quality/domain terms; false positives/negatives with ambiguous text. |

**Gaps Identified:**

1. Most existing approaches rely on rule-based or keyword-based methods (e.g., ARM tool) which detect structural and linguistic issues but fail to capture semantic meaning and correctness of requirements.
2. Current ML/NLP techniques improve extraction and classification but are highly dependent on domain-specific datasets and cannot generalize well across different projects or domains.
3. Many automated systems focus only on a single stage of requirement engineering (extraction, validation, or contradiction detection) but do not provide a complete end-to-end framework.
4. Approaches for requirement validation and contradiction detection often suffer from false positives/negatives, especially when dealing with ambiguous or context-heavy language.
5. Existing tools face difficulty in handling multi-format documents (PDF, Word, Excel, emails) and unstructured inputs, which are very common in real projects.
6. Automated methods lack seamless integration with agile tools (like JIRA, Confluence) and industry standards (like IEEE requirement specifications).
7. Some approaches (like scientific-domain extractions) are too domain-specific and lack scalability to other domains.
8. Overall, literature shows progress in automation but no unified, collaborative, and scalable system exists that combines AI/NLP with real-time validation, prioritization, and IEEE-compliant documentation.

**Objectives framed based on Gaps**

1. To enable semantic understanding of requirements using advanced AI/NLP models, overcoming the limitations of rule-based and keyword-based tools.
2. To automate end-to-end requirement engineering from extraction, classification, and validation to documentation instead of focusing on isolated phases.
3. To ensure multi-format document support (Word, PDF, Excel, emails), enabling seamless requirement extraction from diverse sources.
4. To improve requirement accuracy and validation by integrating real-time clarification, contradiction detection, and prioritization methods.
5. To provide a collaborative and scalable platform by integrating with agile tools (JIRA, Confluence) and generating IEEE-compliant requirement documents for industry standards.

**Problem Statement:**

In software development, requirement gathering is often manual, time-consuming, and prone to human error. Information is scattered across documents such as Word, Excel, PDFs, and emails, leading to inconsistencies and missing details. This slows down project initiation, reduces accuracy, and increases costs. There is a strong need for an automated, intelligent system that can extract, classify, validate, and organize requirements into structured, standard-compliant documents while supporting collaboration and traceability.

**Proposed Methodologies:**
The system will leverage AI and NLP models such as BERT for classification of functional and non-functional requirements, GPT-4 for generating structured requirement documents, and Code Llama for code/test case generation. Data extraction libraries such as python-docx, pdfminer, and pandas will handle multi-format document parsing. Integration with JIRA and Confluence will support collaboration and version control. The MoSCoW method will be used for prioritization of requirements.

**a. Software and Hardware Requirements:**
**Software Requirements:**

- **Programming Languages/Frameworks:** Python (for NLP pipelines, document ingestion), TensorFlow/PyTorch (for BERT/GPT-4 integration), Node.js/Java (for backend services).
- **AI/ML Models:** BERT (requirement classification & prioritization), GPT-4 (document generation), Code Llama (test case & code generation).
- **Databases & Version Control:** MySQL / PostgreSQL for requirement storage, Git for version control.
- **Collaboration & Tracking Tools:** JIRA for issue tracking, Confluence for documentation, APIs for integration.
- **Libraries & Tools:** python-docx, pdfminer, pandas (document parsing), REST APIs for JIRA/Confluence.

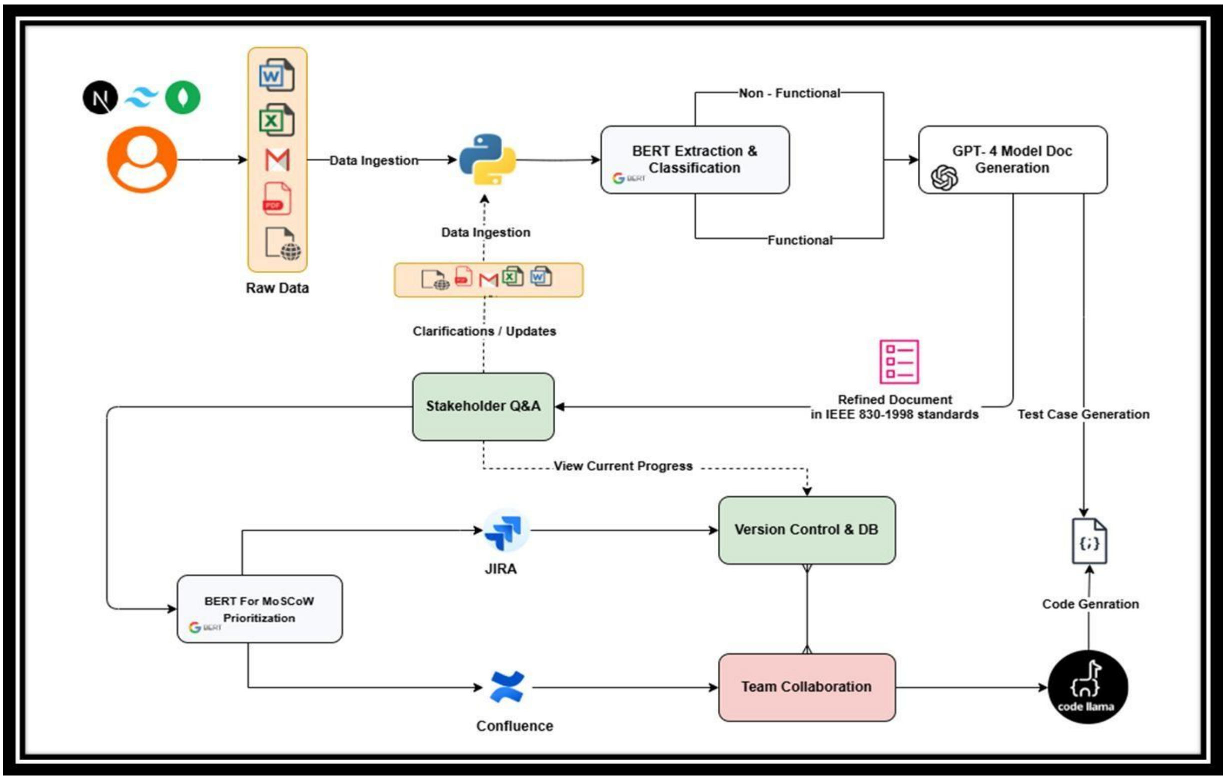
**Hardware Requirements:**

- Development machine with at least 16 GB RAM, multi-core CPU, and GPU (NVIDIA CUDA-enabled) for model execution.
- Cloud environment (AWS/Azure/GCP) for scalability and model deployment.

**b. Algorithms**

1. **Data Ingestion & Parsing Algorithm** – Collects raw data from multiple sources (Word, PDF, Excel, Emails) using parsing libraries.
2. **BERT-based Extraction & Classification Algorithm** – Identifies functional vs. non-functional requirements using transformer embeddings.
3. **GPT-4 Document Generation Algorithm** – Converts classified requirements into IEEE 830:1998 compliant structured documents.
4. **MoSCoW Prioritization with BERT** – Applies NLP-based prioritization (Must-have, Should-have, Could-have, Won't-have).
5. **Stakeholder Q&A Algorithm** – Interactive clarification using fine-tuned LLM for resolving ambiguities.
6. **Test Case & Code Generation Algorithm (Code Llama)** – Generates test cases and initial code skeletons from refined requirements.

**c. Architecture Diagram**




**Expected Outcomes:**

- Automated extraction of requirements from multiple file formats.
- Accurate classification into functional and non-functional requirements.
- IEEE-compliant structured requirement documents.
- Real-time validation and prioritization of requirements.
- Collaboration support with JIRA and Confluence integration.
- Reduced manual effort, improved accuracy, and enhanced project traceability.

**References:**


[1] Vinay, S., Aithal, S., & Desai, P. (2009, March). An approach towards automation of requirements analysis. In *Proceedings of the International MultiConference of Engineers and Computer Scientists* (Vol. 1).

[2] Umar, M. A., & Lano, K. (2024). Advances in automated support for requirements engineering: a systematic literature review. Requirements Engineering, 29(2), 177-207.

[3] Singh, M. (2018, August). Automated validation of requirement reviews: a machine learning approach. In 2018 IEEE 26th International Requirements Engineering Conference (RE) (pp. 460-465). IEEE.

[4] Haris, M. S., & Kurniawan, T. A. (2020, November). Automated requirement sentences extraction from software requirement specification document. In *Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology* (pp. 142-147).

[5] Faisal, M., Issa, G. F., Ayub, I., Asadullah, M., Joiya, U. N., & Iqbal, M. (2022, February). How automate requirements engineering system effects and support requirement engineering. In 2022 International Conference on Business Analytics for Technology and Security (ICBATS) (pp. 1-3). IEEE.

[6] Almusharraf, N., & Alotaibi, H. (2023). An error-analysis study from an EFL writing context: Human and automated essay scoring approaches. Technology, Knowledge and Learning, 28(3), 1015-1031.

[7] Li, Y., Guzman, E., Tsiamoura, K., Schneider, F., & Bruegge, B. (2015). Automated requirements extraction for scientific software. *Procedia Computer Science*, *51*, 582-591.
[8] Wilson, W. M., Rosenberg, L. H., & Hyatt, L. E. (1997, May). Automated analysis of requirement specifications. In *Proceedings of the 19th international conference on Software engineering* (pp. 161-171).
[9] Umar, M. A., Lano, K., & Abubakar, A. K. (2025). Automated requirements engineering framework for agile model-driven development. Frontiers in Computer Science, 7, 1537100.
[10] Gärtner, A. E., & Göhlich, D. (2024). Automated requirement contradiction detection through formal logic and LLMs. Automated Software Engineering, 31(2), 49.