**Current End-to-End Solution Perspective**

Data goes in → Validated AI-Generated Requirements come out

* **Input**: Requirement-related information sources
* Domain knowledge bases (healthcare datasets, medical QA repositories like Banking/MedQuAD in experiments).
* Stakeholder interviews, user stories, and chat logs.
* Existing requirement repositories (PROMISE dataset, PURE, etc.).

**Retrieval-Augmented Generation (RAG) Flow**

**Preprocessing & Chunking**

Text cleaning, segmentation, normalization of requirement artifacts.

**Embedding & Retrieval**

Sentence-BERT generates vector embeddings of requirement-related text.

FAISS index retrieves top-k relevant chunks for any input query.

**Requirement Generation (LLM)**

LLM (FLAN-T5 / GPT-based) generates candidate requirements in standardized format (“The system shall …”).

Requirements mapped to supporting evidence.

**Classification & Validation**

**FR/NFR Classifier (BERT-based)**

Labels requirements as *Functional* or *Non-Functional*.

**Validation Layer**

NLI-based hallucination detection (ensures requirement is entailed by retrieved evidence).

Cosine similarity check between query & generated requirement (appropriateness).

Automatic quality metrics: Precision@k, Recall@k, F1, hallucination rate.

* **Output**: Requirement Assessment & Visual Insights

**Actionable Insights**

**For Developers / Analysts:**

Automatically extract well-structured, validated requirements.

Reduce manual effort and ambiguities in requirement engineering.

**For Researchers:**

Reproducible AI-enabled methodology for requirement specification.

Benchmarkable metrics (retrieval, generation, classification).

**For Stakeholders / Users:**

Transparent requirements with evidence traceability.

Ability to assess system reliability before implementation.

**AI Improvement End-to-End Solution Perspective**

**Input Layer (requirements documents, domain datasets, stakeholder text data) →  
AI Processing Layer (RAG, NLP, ML classification, anomaly detection, clustering, recommendations) →  
Output Layer (validated requirements, quality scores, alerts, summaries, requirement clusters).**

**1. Predictive Requirement Quality Modelling (ML)**

Train ML models on historical requirement datasets (e.g., PROMISE, PURE, Banking Chatbot data) to predict the **quality and correctness** of new requirements.

* **Input:** Requirement attributes (length, ambiguity indicators, similarity to existing requirements, classification features).
* **AI Processing:** Train supervised ML models (Random Forest, Gradient Boosting, Neural Networks).
* **Output:** Probability score that a generated requirement is *clear, testable, and correct*.

**2. Anomaly Detection in Requirement Specifications**

Detect unusual or risky patterns in requirements to flag early signals of **low-quality or conflicting requirements**.

* **Input:** Requirement repository (time-series of revisions, acceptance/rejection logs, classifier scores).
* **AI Processing:** Methods – Autoencoders, Isolation Forest, LSTM anomaly detection.
* **Output:** Alerts such as:
  + “Unusual spike in ambiguous requirements.”
  + “Requirement conflicts detected in NFR set.”

**3. Natural Language Processing (NLP) on Requirement Texts**

Analyse textual content of stakeholder inputs, SRS documents, and generated requirements.

* **Input:** Requirement text, user stories, stakeholder comments.
* **AI Processing:**
  + **Sentiment analysis:** Detect acceptance/resistance trends from stakeholder feedback.
  + **Topic modelling:** Identify recurring requirement themes (usability, performance, security, reliability).
  + **Ambiguity & clarity detection:** Flag vague requirements using linguistic features.
  + Models – BERT, GPT embeddings, LDA/BERTopic.
* **Output:** Requirement health indicators, e.g.,
  + “30% of NFRs are performance related.”
  + “Detected ambiguity in 15% of FRs.”

**4. AI-Assisted Requirement Process Mining**

Beyond discovering requirement engineering workflows, AI predicts **requirement acceptance/rejection** and recommends **optimal elicitation flows**.

* **Input:** Event logs from requirement lifecycle (elicitation, validation, acceptance/rejection, revisions).
* **AI Processing:**
  + Classify requirement traces as “efficient” vs “inefficient.”
  + Predict likelihood of rejection.
  + Recommend stakeholder interaction patterns that improve clarity.
* **Output:** Workflow optimizations, e.g.,
  + “High probability this requirement will be rejected due to ambiguity; suggest rephrasing before review.”

**5. Clustering Requirements by Specification Profiles**

Group requirements into clusters for better **traceability, prioritization, and risk management**.

* **Input:** Requirement features (functional vs non-functional, similarity embeddings, acceptance history).
* **AI Processing:** Unsupervised ML (k-means, DBSCAN, hierarchical clustering).
* **Output:**
  + Clusters of requirements (e.g., security-focused NFRs, usability-related FRs, ambiguous requirements).
  + Dashboards for comparison:
    - “Your project’s NFR set falls into Cluster 2: high ambiguity, low stakeholder acceptance.”