Indian Institute of Technology Bombay

Department of Chemical Engineering

AI-based Decision Support System for Process Risk Management

Udayanraje Patil

Roll No. 23B0407

28 November 2025

Prof. Sandip S. Roy

Prof. Rahul Nabar

Abstract

This report presents the development, implementation, and evaluation of a Large Language Model based decision support system for process safety and emergency management in chemical industries. The system retrieves relevant historical incident data and generates context-specific safety recommendations to assist emergency managers in making rapid, informed decisions during critical incidents.

The system is built using retrieval-augmented generation, where a curated knowledge base of incident case studies serves as the authoritative reference grounding all outputs. By connecting to verified historical data, the system provides recommendations backed by real incidents rather than generic reasoning alone.

The document details the complete technical foundation, system architecture encompassing query processing and semantic search, the structured incident database, and comprehensive evaluation methodology. The system demonstrates the feasibility of LLM-based decision support for safety management when designed with appropriate safety constraints, human oversight, and clear communication of system capabilities and limitations.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

The Critical Role of Emergency Operations in Chemical Industries
Process industries, especially chemical manufacturing plants, are inherently hazardous environments where the potential for accidents can have catastrophic consequences. Major incidents such as the Bhopal disaster in 1984, the Texas City explosion in 2005, and the Flixborough disaster in 1974 underscore the severe human, environmental, and economic impacts of chemical accidents. These incidents reveal the vital importance of effective emergency management systems capable of rapid response, decision support, and resource coordination to mitigate damage.

Traditional emergency response methods rely heavily on pre-determined procedures, human expertise, and manual decision-making processes. However, the complexity and dynamic nature of emergency scenarios often pose significant challenges, including delayed decisions, information overload, and miscommunication. In recent years, advancements in information technology and artificial intelligence have introduced new avenues for enhancing emergency management through real-time data analysis, predictive modeling, and automated decision support systems. Despite these advancements, integrating AI solutions prudently remains critical, given the safety-critical context.

The Promise and Challenge of Large Language Models
Large Language Models (LLMs), such as GPT and similar architectures, have demonstrated remarkable capabilities in understanding and generating human-like text across numerous applications. Their ability to process vast amounts of data, interpret complex language, and provide contextually relevant responses makes them promising tools for supporting emergency operations. For instance, LLMs can assist safety engineers and emergency responders by answering queries related to safety procedures, analyzing incident reports, and summarizing complex technical documents rapidly.

However, deploying LLMs in safety-critical domains introduces specific challenges. These models are known to generate hallucinated or incorrect responses, which could be disastrous if mistaken as factual in a process safety context. Ensuring that the model's outputs are reliable, safe, and aligned with domain knowledge is essential. Moreover, ethical considerations, such as bias, misinformation, and accountability, must be addressed before such models can be widely adopted in real-world emergency scenarios.

Why Now? The Convergence of Needs and Capabilities
The current technological landscape presents a timely convergence of critical needs and advanced capabilities. The increasing volume of operational data from industrial plants, combined with the maturation of LLM technologies, opens up opportunities for creating intelligent decision support systems in process safety domains. Governments, regulatory bodies, and industry stakeholders are seeking innovative solutions to improve safety standards while reducing risks and costs.

Furthermore, recent progress in AI safety research emphasizes the importance of developing controlled, explainable, and trustworthy AI systems for high-stakes domains. This provides both motivation and a framework to explore the integration of LLMs into emergency management processes responsibly. The project aims to harness this confluence of technological readiness and societal need to enhance safety procedures, reduce accident impacts, and support decision-makers during critical moments.

1.2 Problem Statement

The central problem addressed by this project is: *How can Large Language Models be effectively leveraged to create an intelligent decision support system that assists emergency operations in chemical facilities by providing timely, contextually relevant, and safety-conscious guidance?*

Specifically, the project investigates whether an LLM can be configured and augmented with domain-specific knowledge to answer safety-related questions from operators and emergency managers, interpret incident descriptions and recommend appropriate response actions, provide guidance on regulatory compliance and safety procedures, generate concise summaries of complex emergency scenarios, and do so while maintaining accuracy, avoiding unsafe recommendations, and acknowledging its limitations. The knowledge foundation for this capability involves curating a dataset of real and synthetic emergency scenarios, chemical incident case studies, applicable safety regulations and standards, and best-practice procedures. This dataset is processed to create a searchable knowledge base from which relevant information can be retrieved and provided as context to the LLM, ensuring that responses are grounded in reliable, authoritative sources rather than generic internet-derived information.

1.3 Objectives and Scope

1.3.1 Primary Objectives

This project aims to develop a foundational LLM-based decision support architecture that integrates domain-specific safety knowledge through retrieval-augmented generation (RAG) or similar approaches, using currently available tools and models without requiring massive computational resources. The project will prepare and organize a domain-specific knowledge base of chemical process safety information, design and implement effective prompting strategies and safety constraints to guide the LLM toward accurate and appropriately cautious responses, conduct rigorous evaluation using both quantitative metrics and qualitative expert review, identify limitations and risks specific to deploying LLMs in safety-critical emergency contexts, and contribute to the emerging intersection of LLMs and process safety by publishing findings and methodologies.

1.3.2 Scope and Boundaries

This project focuses on developing a working prototype system using Python, publicly available LLM APIs or open-source models, vector database technologies, and a curated knowledge base of chemical safety information. The system will be evaluated on representative safety tasks including Q&A on procedures, incident interpretation, regulatory guidance, and report summarization, with expert review by domain specialists. However, this is an academic research project and does not include full fine-tuning of large models on proprietary industrial datasets, integration with real-time plant monitoring or SCADA systems, formal safety certification or deployment in operational chemical plants, or access to confidential incident databases from industrial facilities. The system is designed to assist trained operators and emergency managers but does not replace expert human judgment, and regulatory requirements vary by jurisdiction requiring adaptation for specific deployment contexts.

1.4 Structure of the Report

This report is organized into eleven chapters progressing from contextual foundations through technical details to evaluation findings and practical implications. Chapter 1 provides the introduction and motivation. Chapter 2 reviews literature on process safety, decision support systems, and AI applications in safety. Chapter 3 explains the technical background of LLMs, prompt engineering, and retrieval-augmented generation. Chapter 4 describes the problem definition, dataset, and preprocessing. Chapter 5 details the system architecture and design. Chapter 6 covers model selection, prompt design, and implementation details. Chapter 7 describes the evaluation methodology including metrics and expert review protocols. Chapter 8 presents results and analysis from testing. Chapter 9 discusses implications and deployment readiness. Chapter 10 addresses safety, ethics, and limitations. Chapter 11 concludes with key findings and suggestions for future research.

For readers seeking to understand the project context and motivation, the Introduction chapter is the starting point. Process safety professionals should focus on Chapters 1-2 and 8-10 for domain-specific insights. Machine learning researchers should emphasize Chapters 3, 5-6, and the appendices for technical details. All readers are encouraged to pay particular attention to Chapter 10 on safety and limitations, as responsible deployment of AI in safety-critical applications requires understanding both capabilities and constraints.

Chapter 2

Literature Review

2.1 Process Safety and Emergency Management in Chemical Industries

Process safety encompasses the application of management systems, engineering practices, and operational discipline to prevent or mitigate unplanned releases of hazardous materials or energy in chemical facilities. Chemical process industries operate under inherent risks due to the hazardous nature of materials processed, extreme operating conditions such as high temperatures and pressures, and complex interdependencies between process systems. Major chemical accidents including the Bhopal disaster (1984), Texas City refinery explosion (2005), and other large-scale incidents demonstrate the catastrophic consequences of inadequate process safety management. Modern process safety frameworks emphasize hazard identification and risk assessment, implementation of engineering controls and design safety factors, administrative controls through procedures and training, emergency preparedness and response planning, and continuous improvement based on incident learning. Emergency operations represent critical phases where rapid, accurate decision-making can significantly reduce losses. During emergency scenarios such as unplanned chemical releases, equipment failures, fires, or loss of containment, emergency managers must make decisions on evacuation planning, resource allocation, on-site and off-site coordination, technical incident management, and regulatory notifications, typically under severe time constraints and uncertainty.

2.2 Applications of Machine Learning and LLMs in Process Safety

Recent advances in machine learning have opened new possibilities for safety applications. Predictive maintenance uses sensor data and machine learning to predict equipment failures before they occur, reducing unplanned downtime and related risks. Anomaly detection identifies unusual process behavior that may indicate emerging risks. Natural language processing extracts safety information from incident reports, manuals, and regulatory documents, enabling systematic analysis of unstructured safety data. Large Language Models represent a powerful new class of AI systems trained on vast amounts of text data, enabling them to understand and generate human language with remarkable capability. Recent LLMs such as GPT-3.5, GPT-4, LLaMA, and Claude have demon-

strated capabilities in question-answering, summarization of long documents, information extraction and categorization, dialogue and multi-turn conversation, and reasoning over domain-specific knowledge when provided appropriate context. In process safety contexts, LLMs offer potential advantages including natural language interfaces that allow operators to pose questions without learning complex systems, flexibility to adapt to varied scenarios through prompt adjustment rather than rule re-engineering, capability to synthesize information from multiple sources via retrieval-augmented generation, and explainability since outputs are in natural language making recommendations more interpretable. However, deploying LLMs in safety-critical domains raises significant concerns: hallucinations where LLMs generate plausible-sounding but factually incorrect information, overconfidence in presenting uncertain information, lack of real-world grounding as LLMs cannot directly sense plant conditions, safety alignment risks if systems suggest unsafe actions, and questions of accountability and liability. These challenges necessitate careful system design, rigorous evaluation, and human oversight when considering LLM deployment in safety-critical applications.

2.3 Decision Support Systems for Emergency Operations

A Decision Support System (DSS) is a computational tool designed to help decision-makers use data, information, models, and domain knowledge to solve complex problems and make decisions more effectively. In emergency management contexts, DSS are critical because they can rapidly synthesize large amounts of information, apply structured logic to scenarios, and provide recommendations that augment human expertise during high-pressure situations where time is limited and consequences are severe.

Traditional DSS in chemical emergency management have typically employed rule-based or model-based approaches. Rule-based systems encode expert knowledge via explicit conditional logic (e.g., "IF pressure exceeds X AND temperature exceeds Y AND relief valve fails, THEN evacuate Zone A"). Model-based systems use consequence prediction models to calculate impact zones, estimate affected populations, and predict dispersion of hazardous materials. These systems often integrate with data sources such as Geographic Information Systems (GIS) or sensor networks to provide situational awareness. Examples include CAMEO (Computer-Aided Management of Emergency Operations) for predicting consequence zones of chemical releases, ALARA (As Low As Reasonably Achievable) frameworks that guide decision-making to minimize risk exposure, and risk-based regulatory systems that inform investment prioritization in safety infrastructure.

However, traditional DSS suffer from significant limitations that reduce their effectiveness in modern emergency operations. Current approaches to emergency decision support have several critical gaps that this project addresses:

2.4 Gaps in Existing Approaches

Accessibility limitations: Rule-based DSS require substantial training to use effectively. Operators must learn complex interfaces, understand system logic, and navigate multiple menus or command structures. Under emergency stress, operators may not consult these systems, preferring instead to rely on memory or instinct, leading to sub-

optimal decisions. The complexity creates a barrier to use precisely when the system is most needed.

Contextual inflexibility: Predefined procedures and rules are designed for anticipated scenarios but may not apply perfectly to novel or unusual combinations of circumstances. When an incident deviates from expected patterns, the system may offer irrelevant guidance or fail to provide any recommendations. This brittleness is particularly problematic in chemical emergencies where unique facility configurations, chemical combinations, or environmental conditions can create unprecedented scenarios.

Limited integration with modern NLP: Few traditional DSS leverage state-of-the-art natural language processing and modern artificial intelligence capabilities. Most require operators to input structured data (selecting from menus, entering codes) rather than communicating in natural language. This constraint reduces accessibility and slows response time during emergencies when operators should be able to describe situations intuitively.

Knowledge management challenges: Updating traditional DSS with new regulations, emerging best practices, or lessons learned from recent incidents is cumbersome and expensive. Rules must be manually modified, tested, and redeployed. Organizations often lag in incorporating new knowledge, meaning their systems may guide operators based on outdated practices or obsolete regulations.

Limited research intersection: While extensive work exists on both LLMs and process safety independently, limited research explores how Large Language Models can be carefully integrated with safety-domain knowledge to create more intelligent decision support systems for chemical emergency operations. This project addresses this gap by investigating whether LLMs can provide more accessible, flexible, and intelligent augmentation to emergency decision-making.

This project bridges these gaps by investigating how LLMs can be carefully integrated with safety-domain knowledge to create a more accessible, flexible, and intelligent decision support system. The approach maintains human-in-the-loop oversight, grounds recommendations in authoritative knowledge sources, and rigorously evaluates both capabilities and limitations. The goal is not to replace human expertise but to enhance and accelerate expert decision-making by providing rapid, natural-language-accessible information synthesis and recommendation generation.

Chapter 3

Technical Background

3.1 Fundamentals of Language Models

A language model is a probabilistic system that assigns probabilities to sequences of tokens (words or subword units). Fundamentally, it learns patterns in text and predicts what word should come next given previous words. Traditional language models (such as n-gram models and early recurrent neural networks) struggled with long-range dependencies in text, making it difficult to capture meaningful relationships between distant words separated by many tokens. This was a critical limitation because understanding language often requires knowing what was said many sentences or paragraphs earlier.

The breakthrough came with the Transformer architecture (Vaswani et al., 2017), which introduced self-attention mechanisms. Self-attention allows the model to weigh the importance of different tokens regardless of their distance in the sequence. Instead of processing words sequentially and losing earlier context, the Transformer can look at any word and determine its relevance to any other word, no matter how far apart they are. This was revolutionary for language understanding.

The Transformer architecture consists of several key components working together:

- **Tokenization:** Input text is split into smaller units called tokens (individual words or subword pieces). For example, "explosion" might be split into "explo" and "sion".
- **Embeddings:** Each token is converted into a numerical vector representation in high-dimensional space. These vectors capture semantic meaning; similar words have similar vector representations.
- **Self-Attention:** The model computes attention scores between all pairs of tokens, determining which tokens are most relevant to each other. This allows the model to understand relationships and dependencies.
- **Feed-Forward Networks:** Non-linear mathematical transformations are applied to process information further.
- **Encoder-Decoder Stacks:** Multiple layers of attention and transformation operations are stacked, allowing the model to learn increasingly abstract patterns at each layer.

For language generation tasks like our system, decoder-only models (such as the GPT series) use an autoregressive approach: given a sequence of tokens, the model probabilistically predicts the next most likely token, then uses that prediction to generate

the subsequent token, and repeats this process until a complete response is generated or a stopping condition is met. This sequential generation allows the model to produce coherent, contextually appropriate text one token at a time.

3.2 Large Language Models: Scale and Capabilities

Large Language Models (LLMs) are Transformer-based models trained on massive text datasets containing hundreds of billions of tokens, often scraped from the internet and various text sources. The scale is enormous: these models contain billions to hundreds of billions of learnable parameters (mathematical weights and values). This massive scale is crucial because it enables the model to learn very rich, nuanced representations of language and world knowledge.

Key characteristics of LLMs include:

- **Pre-training on Unlabeled Data:** LLMs are trained using self-supervised learning, where the training objective is to predict the next token given previous tokens. No human annotation is required; the model learns directly from raw text. This enables learning from vastly larger datasets than traditional supervised learning approaches.
- **Few-Shot Learning Capability:** Once trained, LLMs can perform new tasks with minimal examples. By providing a few examples in the prompt itself, the model can learn the task pattern without any parameter updates. This is called in-context learning and is a remarkable emergent capability.
- **Generative Capability:** LLMs can generate new text that is coherent, contextually relevant, and often of high quality. They don't just memorize training data but learn to generate novel text following learned patterns.
- **Diverse Capabilities:** LLMs have demonstrated strong performance across diverse tasks including translation between languages, question-answering, summarization of long documents, creative writing, coding, and reasoning about complex topics.

Prominent LLM examples include GPT-3.5 and GPT-4 from OpenAI (proprietary, available via API), open-source LLaMA models from Meta (ranging from 7B to 70B parameters), Mistral models, and Claude from Anthropic (designed with emphasis on safety and alignment). For our emergency decision support system, we selected an open-source model, specifically TinyLLaMA (11B parameters), for several strategic reasons detailed in Section 6.1.

3.3 Model Selection: Why TinyLLaMA?

Our system uses TinyLLaMA, a smaller open-source language model with 11 billion parameters. This choice reflects careful consideration of multiple factors relevant to safety-critical emergency operations:

Computational Efficiency: TinyLLaMA can run on standard hardware without requiring high-end GPUs or cloud infrastructure. This means the system can be deployed

locally on facility computers, improving reliability and reducing dependence on internet connectivity. In an emergency, network failures should not prevent the system from functioning.

Reduced Latency: Smaller models generate responses faster than enormous models like GPT-4. In emergency scenarios where decisions must be made rapidly, response time matters. A 2-5 second response is acceptable; waiting 30 seconds is not.

Cost and Accessibility: As an open-source model, TinyLLaMA is freely available and can be modified if needed. This is important for institutional use where recurring API costs for commercial models (like GPT-4 through OpenAI) would be prohibitive. Researchers and future students continuing this project can use and modify the system without licensing restrictions.

Control and Transparency: Open-source models allow complete visibility into model architecture and weights. For safety-critical applications, this transparency is valuable. We can verify behavior, implement custom safety measures, and understand how decisions are made.

Performance-Tradeoff: While TinyLLaMA has fewer parameters than GPT-4, it still demonstrates strong performance on natural language understanding and generation tasks. For domain-specific safety applications with our knowledge base (RAG system), it provides excellent performance at a fraction of computational cost.

Reliability: By controlling the model locally, we reduce dependency on external services. There is no API rate limiting, no subscription costs, and no risk that a vendor discontinues or changes pricing on the service.

This pragmatic approach prioritizes deployment reality over bleeding-edge performance. A system that works reliably in practice is more valuable than one theoretically more capable but inaccessible in operational environments.

3.4 Prompt Engineering and Retrieval-Augmented Generation

Prompt engineering is the art and science of crafting input text (prompts) that guide LLMs to generate desired outputs. The quality of the prompt directly influences response quality. A well-structured prompt typically includes three components:

1. **System Instruction:** High-level guidance on the model's role and behavior. For example: "You are an expert in chemical process safety and emergency operations. Your role is to provide rapid, accurate guidance to operators during emergency situations. Always prioritize safety above all other considerations."

2. **Context or Background Information:** Relevant documents, procedures, case studies, or examples that inform the response. In our system, this includes retrieved safety procedures and similar incident case studies.

3. **User Query or Task:** The specific question or instruction. For example: "What are the immediate actions if a chlorine gas leak is detected in Area C?"

In-context learning is a remarkable capability where LLMs can learn to perform new tasks by being shown examples within the prompt itself, without any modification to the model's weights or parameters. This enables few-shot adaptation to new domains or

tasks. For example, by providing the model with a few examples of incident descriptions followed by recommended precautions, the model learns the pattern and can apply it to new incidents described in the current query.

Retrieval-Augmented Generation (RAG) addresses a fundamental limitation of standalone LLMs: while they possess broad general knowledge from training data, they may lack depth in specialized domains and cannot access information beyond their training data cutoff date. A standalone LLM asked about recent chemical regulations or specific facility procedures would likely generate plausible-sounding but potentially incorrect information (a phenomenon called "hallucination").

RAG operates through the following mechanism:

1. **Knowledge Base Storage:** A domain-specific knowledge base is created containing incident case studies, safety procedures, regulatory documents, and technical specifications. This is stored in a vector database for efficient searching.

2. **Query Retrieval:** When a user poses a query, the system retrieves the most relevant documents from the knowledge base using semantic similarity search. Instead of keyword matching, semantic search understands meaning, so a query about "pressure relief" would retrieve documents about "relief valve" or "PSV systems" even if those exact words aren't in the query.

3. **Context Construction:** The retrieved documents are incorporated into the prompt along with the user query and system instructions.

4. **Grounded Generation:** The LLM generates a response based on the provided context, system instructions, and query. Crucially, the response is grounded in the retrieved information rather than relying solely on the model's general knowledge.

This approach ensures responses are based on current, verified, domain-specific information rather than generic or outdated information. It dramatically reduces hallucination risk because the model is constrained to information available in the knowledge base. If information is not in the knowledge base, the model can be instructed to state that clearly rather than inventing plausible alternatives.

For our emergency safety system, RAG is essential. It means recommendations are always grounded in established safety procedures and incident learnings, not in the LLM's general (and potentially incorrect) understanding of chemical safety.

Chapter 4

Problem Definition and Dataset

4.1 Use-Case Scenarios

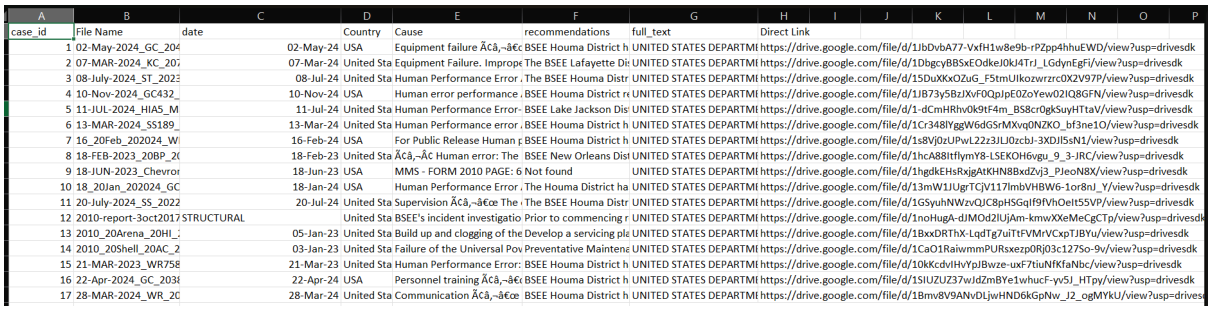
The LLM-based decision support system targets several critical use cases in emergency operations. First, safety procedure question-answering enables operators to ask natural language questions about emergency procedures. When an operator encounters an unfamiliar situation such as "What is the correct procedure if pressure in Reactor A suddenly exceeds the high alarm setpoint?", the system retrieves relevant standard operating procedures, emergency guidelines, and equipment specifications, synthesizes them, and provides a clear, actionable answer grounded in authoritative procedures. Second, incident scenario analysis allows emergency managers to describe complex, active incidents with multiple factors such as chemical type, location, weather conditions, personnel positions, and receive rapid recommendations on evacuation zones, shelter-in-place decisions, resource allocation, containment actions, and notification protocols. Third, regulatory and compliance guidance assists managers in understanding Indian and international regulatory requirements for incident reporting, investigation, and response. Fourth, incident report summarization processes lengthy detailed investigation reports and generates concise executive summaries highlighting key facts, root causes, contributing factors, and lessons learned, accelerating organizational learning. Fifth, risk assessment support helps evaluate severity and frequency characteristics of emerging risks and provides preliminary risk classifications that inform escalation decisions. These use cases collectively address the full spectrum of emergency operations from prevention and preparation through response and recovery.

4.2 Dataset Source and Structure

The project utilizes a curated dataset of chemical incident case studies converted from PDF reports of accident investigations into structured CSV format. The original data source consisted of a collection of detailed PDF incident investigation reports documenting chemical accidents, equipment failures, human errors, and their consequences across various industrial facilities. These PDFs were manually processed and converted into a structured CSV (Comma-Separated Values) format, with each row representing one incident case study.

The dataset contains the following key columns that form the foundation of the system:

- **case_id:** Unique identifier for each incident (1, 2, 3, ..., N)
- **File Name (title):** The name or title of the incident report
- **date:** Date when the incident occurred (e.g., 02-May-2024, 07-May-2024)
- **Country:** Geographic location where the incident occurred (e.g., USA, United States)
- **Cause:** Description of the root cause and contributing factors that led to the incident (e.g., "Equipment failure", "Human error", "Supervision failure")
- **recommendations:** Documented safety precautions and recommended actions based on the incident analysis (e.g., "Improve maintenance procedures", "Implement SIS system", "Enhanced operator training")
- **full_text:** Complete incident description including detailed account of what happened, how the system responded, and what controls were present or absent
- **Direct Link:** URL or reference to the original incident report for verification and further reading



case_id	File Name	date	Country	Cause	recommendations	full_text	Direct Link
1	02-May-2024_GC_204	02-May-24	USA	Equipment failure Ã¢â‚¬	BSEE Houma District h	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1JbDvbA77-VxfH1w8e9b-rPZpp4hhuEWD/view?usp=drivesdk
2	07-MAR-2024_KC_207	07-Mar-24	United Sta	Equipment Failure. Impropε	The BSEE Lafayette Dis	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1DbgcyBB5xEOdkeJ0k4TrJ_LGdynEgFi/view?usp=drivesdk
3	08-July-2024_ST_2023	08-Jul-24	United Sta	Human Performance Error ,	The BSEE Houma Distr	UNITED STATES DEPARTMI	https://drive.google.com/file/d/15DuXKxOZuG_F5tmUIkozwrzrc0X2V97P/view?usp=drivesdk
4	10-Nov-2024_GC432_	10-Nov-24	USA	Human error performance ,	BSEE Houma District re	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1JB73y5BzJXvF0QpJpE0ZoYew02IQ8GFN/view?usp=drivesdk
5	11-JUL-2024_HIA5_M	11-Jul-24	United Sta	Human Performance Error-	BSEE Lake Jackson Dis	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1-dCmHRhv0k9tF4m_BS8cr0gkSuyHTtaV/view?usp=drivesdk
6	13-MAR-2024_SS189_	13-Mar-24	United Sta	Human Performance error ,	BSEE Houma District h	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1Cr348IYggW6dGSrMXvq0NZKO_bf3ne1O/view?usp=drivesdk
7	16_20Feb_202024_W	16-Feb-24	USA	For Public Release Human p	BSEE Houma District h	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1s8Vj0zUPwL22z3JLJ0zcbJ-3XDJI5sN1/view?usp=drivesdk
8	18-FEB-2023_20BP_2(	18-Feb-23	United Sta	Ã¢â‚¬Â human error: The	BSEE New Orleans Dist	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1hcA88ItFlymY8-LSEKOH6vgu_9_3-JRC/view?usp=drivesdk
9	18-JUN-2023_Chevror	18-Jun-23	USA	MMS - FORM 2010 PAGE: 6	Not found	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1hgdkEHsRxjgAtKHN8BxdZvj3_PJeoN8X/view?usp=drivesdk
10	18_20Jan_202024_GC	18-Jan-24	USA	Human Performance Error ,	The Houma District ha	UNITED STATES DEPARTMI	https://drive.google.com/file/d/13mW1JUgrTCjV117lmbVHBW6-1or8nJ_Y/view?usp=drivesdk
11	20-July-2024_SS_2022	20-Jul-24	United Sta	Supervision Ã¢â‚¬â€œ The	The BSEE Houma Distr	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1GSyuhNWzvQJC8pHSGqIf9fVhOeIt55VP/view?usp=drivesdk
12	2010-report-3oct2017	STRUCTURAL	United Sta	BSEE's incident investigatio	Prior to commencing r	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1noHugA-dJMOd2lUjAm-kmwXXeMeCgCTp/view?usp=drivesdk
13	2010_20Arena_20HI_:	05-Jan-23	United Sta	Build up and clogging of the	Develop a servicing pla	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1BxxDRThX-LqdTg7uiTtFVMrVCxpTJBYu/view?usp=drivesdk
14	2010_20Shell_20AC_2	03-Jan-23	United Sta	Failure of the Universal Pov	Preventative Maintena	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1CaO1RaiwmmPURsxezp0Rj03c127So-9v/view?usp=drivesdk
15	21-MAR-2023_WR758	21-Mar-23	United Sta	Human Performance Error:	BSEE Houma District h	UNITED STATES DEPARTMI	https://drive.google.com/file/d/10kKcdvIHvYpJBwze-uxF7tiuNfKfaNbc/view?usp=drivesdk
16	22-Apr-2024_GC_203	22-Apr-24	USA	Personnel training Ã¢â‚¬	BSEE Houma District h	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1SIUZUZ37wJdZmBYe1whucF-yv5J_HTpy/view?usp=drivesdk
17	28-MAR-2024_WR_20	28-Mar-24	United Sta	Communication Ã¢â‚¬â€œ	BSEE Houma District h	UNITED STATES DEPARTMI	https://drive.google.com/file/d/1Bmv8V9ANvDLjwHND6kGpNw_J2_ogMYkU/view?usp=drivesdk

Figure 4.1: Data Set

The dataset includes real incident cases from chemical manufacturing plants, storage facilities, and transportation scenarios across multiple years (2010-2024) and countries, providing comprehensive coverage of different failure modes and incident types. Each incident represents a learning opportunity, documenting what went wrong, why systems failed, and what precautions should have been in place.

4.3 Data Processing Pipeline

The dataset undergoes a multi-stage processing pipeline to transform raw data into a format suitable for the LLM-based decision support system:

4.3.1 Stage 1: Data Conversion and Cleaning

Raw PDF incident reports were manually processed and converted into structured CSV format. Text normalization procedures remove special characters, standardize formatting, and convert text to consistent case conventions. Data fields are validated for completeness; entries with missing critical information (especially the "Cause" and "recommendations" columns) are flagged. Date fields are standardized to consistent date formats (DD-MMM-YYYY). Geographic information is standardized to country names.

4.3.2 Stage 2: Semantic Embedding Generation

Once the CSV data is cleaned and validated, a critical processing step occurs: converting text fields into mathematical vector representations called embeddings. This process enables semantic similarity search, which is essential for the system to work effectively.

How Embeddings Work: The system uses a pre-trained sentence transformer model (specifically, *all-MiniLM-L6-v2*) to convert text into numerical vectors. Rather than treating text as isolated characters or words, embeddings capture semantic meaning. Words or phrases with similar meanings have similar vector representations. For example:

- The phrase "relief valve failure" and "PSV system malfunction" would have similar embeddings because they represent similar concepts
- "Equipment failure" and "Human error" would have different embeddings reflecting their different semantic meanings
- "Pressure exceeded setpoint" and "High pressure alarm triggered" would have similar embeddings despite different wording

Mathematically, embeddings are typically 384-dimensional vectors (numbers), meaning each incident's text is converted into 384 numbers that numerically encode its meaning. The embedding model learns these representations during pre-training on millions of text examples.

Which Fields Get Embedded: The system creates embeddings from combining the most informative text fields:

$$\text{Embedding Input} = \text{title} + \text{cause} + \text{full\_text}$$

This combined representation captures the incident title, root cause, and detailed description, providing rich context for similarity matching.

4.3.3 Stage 3: Vector Indexing with FAISS

After embeddings are generated, they are indexed in a vector database using FAISS (Facebook AI Similarity Search). FAISS is an efficient similarity search library that allows rapid finding of similar incidents.

How Vector Indexing Works: Traditional database searches use keyword matching ("Find all records containing the word explosion"). Vector-based search is fundamentally different:

1. User enters a query (e.g., "A chlorine gas leak occurred in storage area")
2. System converts the query to an embedding using the same sentence transformer model
3. System computes similarity between the query embedding and all incident embeddings in the index
4. Incidents are ranked by similarity score (typically 0 to 1, where 1.0 is identical)
5. Top-K most similar incidents are retrieved (typically K=3)

Why This Matters: Consider a user query: "What should we do if pressure equipment ruptures?" A keyword-based system might miss incidents titled "Failed vessel in reactor section". A semantic search would recognize that vessel rupture and pressure equipment failure are semantically similar concepts and retrieve the relevant incident. This semantic understanding is crucial because emergency operators might describe situations using different terminology than the incident database, yet the concepts are identical.

4.3.4 Stage 4: Two-Model Precaution Generation

A key innovation in this system is the use of **two different models** for generating safety precautions:

Model 1: Dataset-Based Precautions (Direct Retrieval)

When similar incidents are retrieved from the database, the system can directly extract the "recommendations" column from those incidents. These are real, documented precautions that were recommended after investigating actual accidents.

Process:

1. User query is converted to embedding
2. Similar incidents are retrieved (top 3 most similar)
3. System extracts the "recommendations" field from each retrieved incident
4. These proven recommendations are presented to the user

Advantage: These recommendations are grounded in real incident data. They represent actual safety lessons learned from past accidents. They have been vetted by safety professionals during incident investigation.

Model 2: AI-Generated Precautions (LLM Generation)

In addition to extracting recommendations from similar incidents, the system uses TinyLLaMA to generate novel precautions based on the specific incident scenario.

Process:

1. User query and retrieved similar incidents are provided as context
2. LLM is prompted: "Based on these similar incidents, list 5 critical precautions the operator must take immediately"
3. TinyLLaMA analyzes the context and generates contextually appropriate precautions
4. Generated precautions are validated against the "full_text" of retrieved incidents to ensure consistency

Advantage: AI-generated precautions can adapt to novel aspects of the current scenario that may not exactly match historical incidents. The LLM can reason about chemical properties, facility types, and systemic vulnerabilities to suggest precautions tailored to the specific situation.

Hybrid Approach: By providing both dataset-based and AI-generated precautions, the system offers:

- **Proven recommendations** from the dataset backed by real incident history
- **Adaptive recommendations** from the LLM that address specific scenario details
- **Redundancy and verification:** Users can compare both sets of recommendations to validate consistency
- **Flexibility:** If similar incidents exist in the database, use them; if scenario is novel, rely on LLM reasoning

4.3.5 Stage 5: Risk Quantification

As detailed in Chapter 6, the system calculates two key metrics displayed to the user:

Frequency Score (1-5): Calculated using LOPA methodology based on how many similar incidents were found in the database and how recent they are. More recent similar incidents indicate higher frequency.

Consequence Score (1-5): Calculated through keyword analysis of the incident description (e.g., "explosion" $\rightarrow$ score 5, "leak" $\rightarrow$ score 3).

These scores are combined in the Risk Matrix to produce a final risk category (LOW, MEDIUM, SERIOUS, HIGH, CATASTROPHIC).

4.4 Data Organization and Storage

The processed dataset is organized as follows:

- **CSV File (Processed Data):** Contains the structured incident records with all columns listed above
- **Embedding Vectors:** Generated embeddings are stored separately, indexed by case_id for fast retrieval
- **FAISS Index:** Vector database index enabling rapid semantic similarity search across all incidents
- **Metadata Mapping:** Maintains linkage between embeddings and original data so that retrieved incidents can be looked up and their full information displayed

This organized structure enables rapid retrieval (typically 100-500ms to find top-3 similar incidents) while maintaining full traceability back to original incident data and sources. Users can view original incident details, causes, and recommendations alongside AI-generated guidance, maintaining transparency about where information comes from.

Chapter 5

System Architecture and Design

5.1 High-Level System Overview

The LLM-based decision support system follows a retrieval-augmented generation (RAG) pipeline architecture that integrates TinyLLaMA with a domain-specific knowledge base of chemical incident data. At its core, the system helps emergency operators and managers make rapid, informed decisions by combining historical incident learning with intelligent AI analysis.

When an operator encounters an emergency situation such as a chlorine leak with uncertain consequences, they describe it in natural language through a web interface. The system then searches through historical incidents to find similar past cases, learns from what happened and what was recommended, and provides the operator with actionable safety precautions. What makes this system unique is that it provides recommendations in two forms: proven recommendations extracted from similar historical incidents, and intelligently generated recommendations from an AI model that adapts to the specific scenario details.

The overall workflow consists of ten key stages. First, a user poses a query through a Gradio-based web interface. The system then processes this query, converting it into a mathematical representation that enables comparison with historical data. Next, it searches a database of thousands of chemical incidents to find the most similar cases. From these retrieved cases, the system calculates risk scores using LOPA methodology. It then generates precautions using two independent models: extracting recommendations from similar incidents and generating new recommendations using TinyLLaMA AI. The system constructs a comprehensive prompt, processes it through TinyLLaMA, validates the output for safety, and finally presents results to the user via the web interface with risk scores, similar cases, and recommendations.

This architecture ensures that LLM outputs are grounded in verified historical data rather than generic knowledge, significantly reducing hallucination risks. The dual-model approach provides both proven and adaptive guidance, making the system reliable for both common and unusual scenarios.

5.2 System Components and Their Functions

The system comprises six interconnected components that work together seamlessly. Each component has a specific role in transforming an emergency query into actionable recom-

mendations.

5.2.1 Query Processing Module: Understanding the Emergency

When an emergency operator submits a query like "Chlorine gas detected in Area B, wind speed 15 km/h toward residential area, 20 people nearby," the Query Processing module springs into action. This module performs three critical tasks:

- **Text Normalization:** Standardizes the input by converting to consistent case, removing extra spaces, and cleaning special characters
- **Keyword Extraction:** Identifies critical information such as chemical type (chlorine), location (Area B), weather conditions (wind speed), and affected population size
- **Semantic Encoding:** Converts the entire query into a 384-dimensional numerical vector using a pre-trained machine learning model called *all-MiniLM-L6-v2*

Why convert text to numbers? Because the next stage of the system works with vectors, not text. Vector representations capture semantic meaning. Two phrases with similar meanings will have similar numerical representations. This allows the system to find incidents that might use different words but describe fundamentally similar situations. For example, "pressure relief failure" and "PSV malfunction" refer to very similar concepts and will have similar vectors, enabling the system to retrieve both when searching for pressure equipment problems.

5.2.2 Knowledge Base and Vector Index: Learning from History

The knowledge base is the system's memory of past incidents. It contains a CSV dataset with chemical incident information organized into eight structured columns:

- Case identifier and incident title
- Date of occurrence (2010-2024 historical coverage)
- Geographic location (country information)
- Root cause analysis
- Documented recommendations from investigation
- Full incident description and narrative
- Direct links to original investigation reports

This dataset originated from a substantial manual effort converting PDF reports of chemical accident investigations into structured CSV format. The dataset spans from 2010 to 2024 and includes incidents from various countries and chemical types, providing diverse learning examples.

Each incident's textual information (title, cause, and full description) is converted into a numerical vector using the same process as query processing. All these incident

vectors are indexed in FAISS (Facebook AI Similarity Search), a specialized database enabling rapid semantic searching. This indexing enables the system to search through thousands of incidents and find the most similar ones to any new query.

5.2.3 Retrieval and Similarity Search: Finding Relevant Cases

When the user's query vector has been created, the Retrieval module springs into action. The module computes how similar the query vector is to each of the thousands of incident vectors in the FAISS index. This similarity computation happens remarkably fast—typically in 50-200 milliseconds even with thousands of incidents. The system identifies the top 3 most similar incidents and retrieves their full information from the CSV dataset.

Why this approach? Because the recommendations are grounded in real, documented incidents. If an operator's situation matches a past incident, the system can say "Here's what we learned from a very similar situation that happened before." This is far more reliable than an AI model inventing recommendations from general knowledge.

5.2.4 Risk Quantification: Assessing Severity

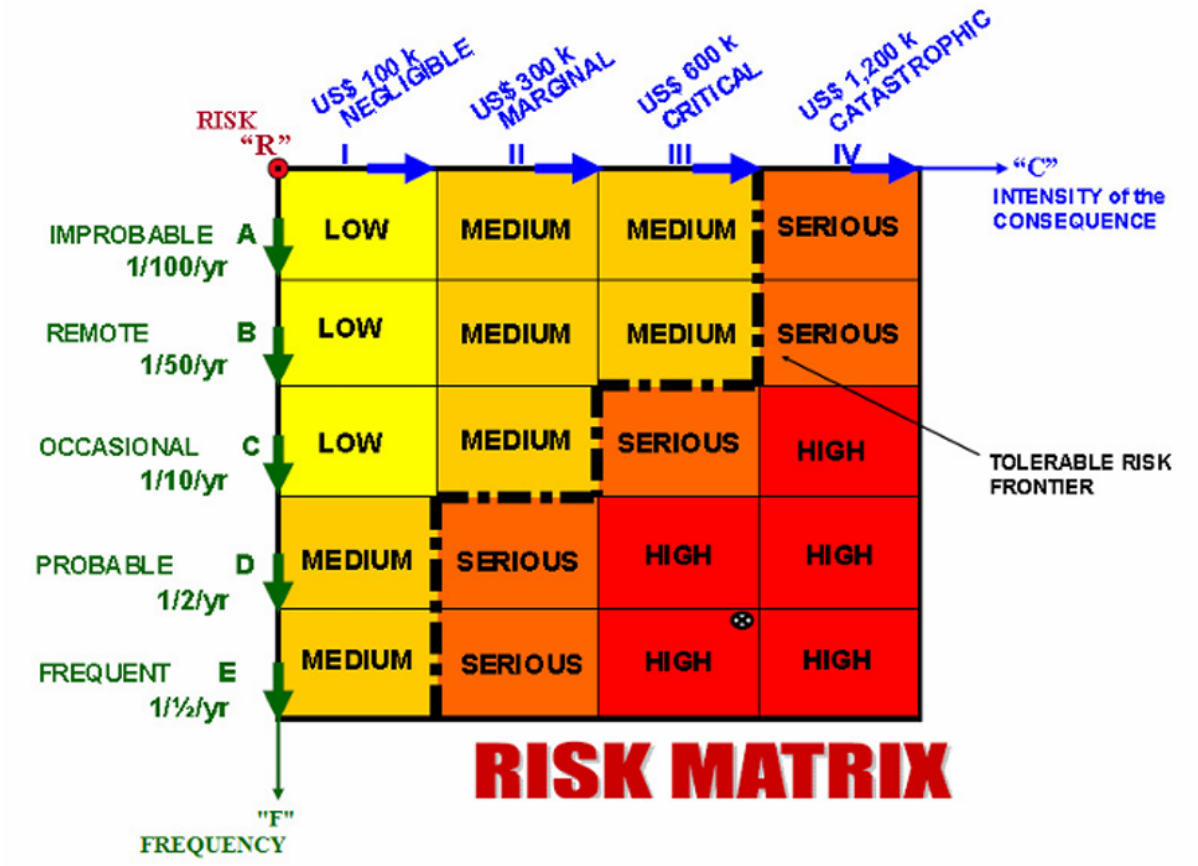


Figure 5.1: User Interface

Based on the three retrieved incidents, the system calculates two critical numbers: a Frequency score (how often such incidents occur) and a Consequence score (how severe

the outcome is). These scores are combined in a Risk Matrix to determine the risk level: LOW, MEDIUM, SERIOUS, HIGH, or CATASTROPHIC.

The Frequency score is calculated using LOPA (Layer of Protection Analysis) methodology by examining how many similar incidents occurred historically and what protection layers were present. The Consequence score uses keyword analysis of the query—words like "explosion" indicate catastrophic consequences (score 5), while "leak" indicates serious but less severe consequences (score 3). This risk quantification is detailed fully in Chapter 6 Section 6.4. The key point is that risk scores are generated through established industry methodology rather than arbitrary scoring, providing credibility and consistency.

5.2.5 Dual-Model Precaution Generation: Two Paths to Safety

Here lies the system's key innovation: precautions are generated through two independent approaches that complement each other.

Model 1 - Dataset-Based Precautions: The system extracts the "recommendations" column from each of the three retrieved incidents. These are real, documented safety recommendations that emerged from actual incident investigations. They carry significant weight because they represent lessons learned from actual accidents—they were recommended by safety professionals who investigated what went wrong and what should have been done differently.

Model 2 - AI-Generated Precautions: Simultaneously, TinyLLaMA analyzes the retrieved incidents and the current query to generate adaptive precautions. The LLM is given a prompt explaining its role, the context of similar incidents, and the specific task. The LLM processes this information and generates recommendations tailored to the specific aspects of the current situation.

By presenting both approaches, operators receive proven guidance backed by incident history, plus intelligent adaptive guidance that considers scenario specifics. Cross-validation between models provides confidence: if both recommend similar precautions, confidence is high. If they differ, operators can apply critical judgment.

5.2.6 Output Presentation and User Interface

The system presents results through a web-based interface built with Gradio, requiring no special software installation. The interface displays comprehensive information including risk categories with color coding, numerical scores, Risk Matrix visualization, retrieved historical incidents with full details, precautions from both generation models, and supplementary analysis such as control layer information.

5.3 Complete System Workflow

The complete system workflow follows this integrated sequence, where each stage builds upon the output of the previous stage:

```
USER INPUT (Gradio Interface)
    ↓
QUERY PROCESSING
  • Normalize and validate text
```

- Extract key concepts (chemical, location, hazard level)
- Generate semantic embedding (384-dim vector)

↓

SEMANTIC SIMILARITY SEARCH (FAISS Index)

- Compute similarity between query and all incidents
- Rank incidents by relevance score
- Retrieve top-3 most similar incidents with full data

↓

RISK QUANTIFICATION (LOPA Methodology)

- Calculate Frequency score (IEF × PFD formula)
- Calculate Consequence score (keyword analysis)
- Determine Risk Category (LOW to CATASTROPHIC)
- Generate Risk Matrix visualization/heatmap

↓

DUAL PRECAUTION GENERATION

MODEL 1: Extract recommendations from dataset
(proven, historically verified)

MODEL 2: TinyLLaMA AI generation
(adaptive to scenario specifics)

↓

PROMPT CONSTRUCTION

- System instruction (role & constraints)
- Retrieved incident context (similar cases)
- User query details
- Generation instruction (5 precautions needed)

↓

LLM PROCESSING (TinyLLaMA via Ollama)

- Submit prompt to localhost:11434
- Process through TinyLLaMA model
- Maintain temperature=0 for deterministic output
- Generate contextually appropriate precautions

↓

POST-PROCESSING & VALIDATION

- Check consistency with source documents
- Apply safety filters (flag unsafe recommendations)
- Calibrate confidence expressions
- Format response for display

↓

OUTPUT ASSEMBLY

- Compile risk scores and category
- Create Risk Matrix heatmap visualization
- Include similar cases with case IDs and links
- Combine precautions from both models
- Generate control layer analysis charts
- Add source citations and references

↓

USER OUTPUT (Gradio Interface)

- Display all results in organized layout

- `Provide clickable links to original reports`
- `Enable follow-up queries`
- `Log query for system monitoring`

The system maintains comprehensive logging throughout this workflow. Every stage records relevant information: which queries were asked and when, which incidents were retrieved and their similarity scores, what risk scores were calculated, what recommendations were generated (from both models), system performance metrics (response time, processing latency), and any errors or safety filter triggers. This logging serves multiple purposes: enabling post-incident analysis, supporting continuous system improvement, and creating audit trails for accountability in safety-critical operations.

5.3.1 System Integration Points

The system integrates with several external technologies and libraries:

- **TinyLLaMA via Ollama:** Local large language model server running at `http://localhost:114` ensuring local processing and data privacy
- **Sentence Transformer:** *all-MiniLM-L6-v2* model for generating semantic embeddings of incident text and queries
- **FAISS Library:** Vector similarity search engine for rapid retrieval of similar incidents from thousands of records
- **Data Processing:** Pandas for CSV data handling, NumPy for numerical operations and matrix calculations
- **Web Interface:** Gradio framework for creating accessible web-based user interface
- **Visualization:** Matplotlib for generating Risk Matrix heatmaps and control layer analysis charts

This modular design enables flexibility and maintainability. If a better embedding model becomes available, only the embedding module needs updating. If TinyLLaMA is replaced with a different LLM, only the LLM interface needs modification. The overall system architecture remains stable and extensible for future improvements.

Chapter 6

Model Development and Implementation

6.1 Model and Framework Selection

The project selects an LLM based on several criteria balancing capability, accessibility, and computational feasibility. For systems with API access, GPT-3.5 or GPT-4 from OpenAI provide state-of-the-art capabilities in understanding context, following complex instructions, and reasoning over sophisticated information. However, these proprietary models come with significant drawbacks for safety-critical applications: they require continuous internet connectivity and API subscriptions, have unpredictable latency depending on external service load, involve recurring costs, and raise concerns about data privacy when queries are sent to external servers.

For systems requiring local deployment or open-source flexibility, LLaMA models from Meta or Mistral models offer strong performance with reasonable computational requirements. Our project selected TinyLLaMA (11 billion parameters), an open-source model, based on careful consideration of multiple factors. TinyLLaMA can run efficiently on standard hardware without requiring high-end GPUs, ensuring accessibility for facility-level deployment. It generates responses in 2-5 seconds, acceptable for emergency operations where every second matters. As an open-source model, it is freely available without recurring API costs. Most importantly for safety-critical applications, it operates locally, maintaining data privacy and ensuring operation even if network connectivity is lost during emergencies.

The supporting technology stack includes Python as the primary programming language, Jupyter notebooks for development and experimentation, FAISS for vector database management enabling rapid semantic search, sentence-transformers library for generating semantic embeddings, Pandas for CSV data manipulation, NumPy for numerical operations, and Gradio for creating the web-based user interface. This technology stack is chosen to maximize accessibility, reproducibility, and ease of modification for future research and system improvements.

6.2 Prompt Design and Safety Constraints

System prompts are carefully crafted to guide TinyLLaMA toward safe, accurate, domain-focused responses. A typical system prompt instructs the model to act as an expert in

chemical process safety and emergency operations, prioritize human safety above all other considerations, base all recommendations exclusively on the provided safety guidelines and retrieved incident data, clearly indicate when information is uncertain or when expert review is needed, never provide recommendations conflicting with established safety standards, and explicitly state when relevant information is unavailable or outside the knowledge base coverage.

Task-specific prompts provide additional guidance for particular types of queries. For incident analysis, the prompt requests structured output with severity assessment and recommended actions. For the precaution generation task, the prompt provides the query context along with the 3 most similar retrieved incidents and instructs the model to "Generate 5 critical safety precautions that an operator must take immediately, considering the hazards identified in these similar incidents and the current scenario." This guides the AI toward practical, actionable recommendations grounded in historical context.

Safety guardrails are implemented at multiple levels. Prompt-level instructions explicitly constrain the model's behavior. Post-processing filters detect and flag potentially unsafe recommendations such as those suggesting evacuation without proper warning procedures or recommending actions that conflict with chemical properties. Output filters check for consistency with source documents to ensure AI-generated recommendations align with the retrieved incident data. Confidence calibration techniques instruct the model to express appropriate uncertainty using phrases such as "Based on the available information" or "This requires expert verification." These guardrails ensure that the system enhances human judgment rather than replacing it, and that recommendations carry appropriate confidence qualifications.

6.3 Implementation Details

The system is implemented in Python using Jupyter notebooks for development and experimentation. The codebase is organized into clearly defined modules, each with specific responsibilities. A data loader module reads and preprocesses the CSV incident dataset, handling missing values, standardizing formats, and preparing data for subsequent processing. An embeddings module generates 384-dimensional semantic vectors for incident texts and queries, managing the vector database through FAISS indexing. A retrieval module executes similarity searches against the FAISS index to find the top-3 most similar incidents. A prompt manager constructs comprehensive prompts by combining system instructions, retrieved incident context, user queries, and generation directives. An LLM interface module communicates with TinyLLaMA via Ollama, submitting prompts to the local server at `http://localhost:11434/api/generate` and receiving generated responses. A post-processor module validates outputs, applies safety filters, checks consistency with source documents, and formats responses for user presentation.

The user interface is implemented using Gradio, a Python library that enables quick creation of web-based interfaces for machine learning models. Gradio eliminates the need for custom frontend development; the interface is defined programmatically in Python, automatically generating a responsive web interface accessible through any browser. Users input queries in a text field, and results including risk scores, similar cases, and precautions are displayed in an organized layout with visualizations.

The system maintains comprehensive logging at every stage of processing. Query logs record what questions operators asked and when. Retrieval logs track which incidents

were retrieved and their similarity scores. Risk assessment logs record calculated Frequency and Consequence scores. Generation logs track precautions produced by both models. Performance logs record response times and processing latencies. Error logs capture any safety filter triggers or anomalies. This comprehensive logging enables post-incident analysis, continuous system monitoring, and provides audit trails for accountability in safety-critical operations.

The code is structured to be modular and extensible. Researchers can swap the embedding model if a better one becomes available, replace TinyLLaMA with a different LLM if desired, modify FAISS settings for different retrieval characteristics, or update safety filter rules based on operational experience. This architecture supports long-term evolution and improvement of the system.

6.4 Risk Quantification: Frequency and Consequence Calculation

The system employs a LOPA-based (Layer of Protection Analysis) risk quantification methodology to calculate the numerical risk scores displayed in the user interface. Understanding how these numbers are generated is essential for interpreting system recommendations and appreciating the scientific foundation of the risk assessment.

6.4.1 How Consequence Scores (1-5) are Generated

The consequence severity is determined through keyword analysis of the incident description. The system maintains a dictionary mapping incident-related keywords to consequence severity scores from 1 to 5 (catastrophic). Specific words are associated with severity levels:

- **Score 5 (Catastrophic):** Keywords such as “fatality”, “death”, “explosion”, “rupture”, “catastrophic”
- **Score 4 (High):** Keywords including “fire”, “blast”, “major release”, “vapor cloud”
- **Score 3 (Serious):** Keywords such as “injury”, “leak”, “spill”, “damage”, “shutdown”
- **Score 2 (Minor):** Keywords including “near miss”, “failure”, “deviation”, “stuck”
- **Score 1 (Low):** Keywords such as “alarm”, “alert”, “warning”, “maintenance”

The system scans the input query text, identifies matching keywords, and assigns the highest corresponding severity score. If the query mentions multiple severity indicators, the maximum score is selected, reflecting the principle of assessing risk based on the worst-case outcome described. For example, a query describing “a chlorine gas leak from a storage tank with potential for explosion” would trigger the “explosion” keyword, resulting in Consequence Score = 5.

6.4.2 How Frequency Scores (1-5) are Generated

Frequency calculation is more complex, using a three-step LOPA-based process:

Step 1: Initiating Event Frequency (IEF) Calculation

The IEF represents how frequently similar incidents have occurred historically. Rather than assuming a fixed value, the system dynamically computes IEF by examining the retrieved historical cases. When the system retrieves similar incidents (typically the top 3 most similar), each is weighted based on its ranking by semantic similarity. A rank-based weighting system allocates higher weights to more similar incidents:

- **Rank 1 (Best match):** Weight = 0.50 (50
- **Rank 2 (Second match):** Weight = 0.33 (33
- **Rank 3 (Third match):** Weight = 0.17 (17

Mathematically, for n similar cases ranked by semantic similarity, the weight for rank i is:

$$w_i = \frac{n - i + 1}{\sum_{j=1}^{n} j} = \frac{n - i + 1}{n(n+1)/2}$$

With three retrieved cases, the denominator is $1 + 2 + 3 = 6$, yielding the weights above. These three weighted cases represent 1.0 total weighted incident count in the historical database (typically 11 years of data, 2014-2025). Therefore:

$$\text{IEF} = \frac{\text{Total Weighted Count}}{\text{11 years}} = \frac{1.0}{11} \approx 0.091 \text{ incidents per year}$$

This approach ensures that the most contextually similar incidents drive the frequency assessment. A recent, highly similar incident contributes more to the frequency score than an older, less similar one.

Step 2: Probability of Failure on Demand (PFD) Calculation

The PFD represents the likelihood that safety systems will fail to prevent the consequence. Industrial facilities have multiple layers of protection (safety barriers), each with an associated failure probability. The system identifies protection layers present in retrieved historical incidents and assigns industry-standard PFD values:

- Safety Instrumented System (SIS, SIL2): PFD = 0.01 (highly reliable)
- Relief Valve or Rupture Disk: PFD = 0.01
- Alarms and Operator Action: PFD = 0.1
- Basic Process Controls (BPCS): PFD = 0.1
- Mitigation (dikes, emergency response): PFD $\approx$ 1.0 (less effective at prevention)

When multiple protection layers exist, their PFDs are multiplied according to LOPA principles, reflecting that redundant systems provide greater protection:

$$\text{Total PFD} = \text{PFD}_1 \times \text{PFD}_2 \times \text{PFD}_3 \times \ldots$$

For example, a system with both SIS (0.01) and relief valve (0.01) would have combined PFD $= 0.01 \times 0.01 = 0.0001$, much more reliable than either layer alone.

The system scans the full text of retrieved incidents for keywords related to protection layers ("SIS", "PSV", "alarm", "controller", "firewater"). When protection layers are identified, the system calculates an aggregated PFD by combining individual PFDs using similarity weights. When no explicit protection layers are mentioned in incident text, the system estimates PFD from incident severity: severe incidents (explosions) are estimated to have PFD ≈ 0.01, moderate incidents (spills, fires) have PFD $\approx 0.1 - 0.4$, and mild incidents (near misses) have PFD $\approx 0.4 - 0.8$.

Step 3: Final Frequency Score and Scaling

The final frequency of consequence (f_C) is calculated using the core LOPA formula:

$$f_C = \text{IEF} \times \text{PFD}$$

This numerical result represents the annual frequency at which the identified consequence could occur. It accounts for both the likelihood of incident initiation (IEF) and the probability that protection layers fail (PFD). The calculated f_C value is then mapped to a 1-5 frequency scale using fixed thresholds:

Frequency Scale	**Description**	**Criteria (f_C value)**
5	Very Frequent	$f_C \geq 0.5$
4	Frequent	$0.1 \leq f_C < 0.5$
3	Occasional	$0.02 \leq f_C < 0.1$
2	Rare	$0.01 \leq f_C < 0.02$
1	Very Rare	$f_C < 0.01$

Table 6.1: Frequency Scale Mapping Thresholds

These thresholds are based on LOPA industry standards and reflect engineering judgment about what frequency ranges justify different risk responses.

6.4.3 Risk Matrix: Combining Frequency and Consequence

Once both frequency (1-5 scale) and consequence (1-5 scale) scores are determined, they serve as coordinates in the LOPA Risk Matrix that produces the final risk category. The risk matrix cross-references these two dimensions to produce intuitive risk categorization: LOW, MEDIUM, SERIOUS, HIGH, or CATASTROPHIC.

The mapping follows LOPA industry standards where higher frequencies and higher consequences combine to produce more severe risk categories. Some key combinations illustrate the logic:

The risk matrix output provides the intuitive risk categorization displayed to users. A scenario with Frequency Scale = 3 and Consequence Scale = 5 results in "HIGH" risk category, triggering more intensive safety recommendations and requiring higher levels of management attention and escalation.

Consequence	Frequency	Risk	Interpretation
5	1	MEDIUM	Catastrophic outcome but very rare—acceptable risk
5	3	HIGH	Catastrophic outcome occurring occasionally—requires action
5	5	CATASTROPHIC	Catastrophic outcome and frequent—unacceptable
1	5	MEDIUM	Minor outcome but very frequent—manages to stay medium
1	1	LOW	Minor outcome, rare—lowest risk

Table 6.2: Risk Matrix Logic: Example Consequence-Frequency Combinations

6.4.4 Worked Example: From Query to Risk Category

Consider a user query describing a chlorine gas explosion scenario. The system processes this as follows:

1. **Consequence Analysis:** The query mentions "explosion", triggering Consequence Score = 5 (catastrophic).
2. **Historical Case Retrieval:** The system retrieves 3 similar historical chlorine/-explosion cases from the knowledge base with similarity scores: Rank 1 (similarity 0.92), Rank 2 (similarity 0.87), Rank 3 (similarity 0.78).
3. **IEF Calculation:** The 3 similar cases with weights 0.50, 0.33, 0.17 represent 1.0 total weighted incidents in 11 years, yielding IEF = $1.0/11 \approx 0.091$ incidents per year.
4. **PFD Assessment:** Analysis of retrieved incidents identifies relief valves (PFD = 0.01) and pressure transmitters (basic control, PFD = 0.1) in the systems. With approximate equal weighting, aggregated PFD ≈ 0.05 (moderate protection layers).
5. **Frequency Calculation:** $f_C = 0.091 \times 0.05 = 0.00455$, falling in range $0.01 \leq f_C < 0.02$, mapping to Frequency Scale = 2 (Rare).
6. **Risk Matrix Lookup:** Frequency = 2 + Consequence = 5 yields "HIGH" risk category.
7. **UI Display:** The interface displays "Frequency: 2/5", "Consequence: 5/5", "Risk Category: HIGH", along with precautions drawn from retrieved incidents and AI-generated recommendations.

This quantitative foundation enables consistent, reproducible risk assessments across diverse scenarios. The multi-step LOPA-based approach ensures that risk categories reflect both the potential severity of outcomes and the statistical likelihood based on historical incident patterns, providing transparency into how the numbers shown in the UI are generated.

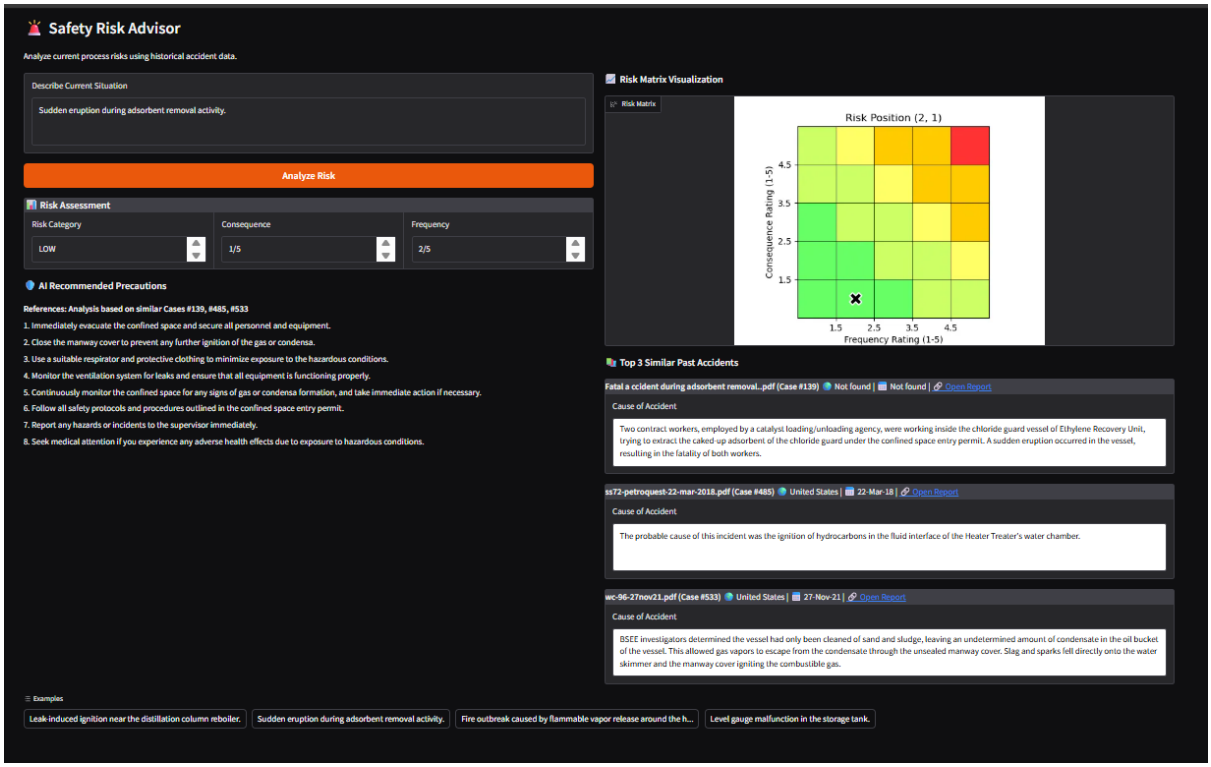


Figure 6.1: User Interface

Chapter 7

Evaluation Methodology

7.1 Evaluation Strategy for the LLM-Based Decision Support System

The LLM-based decision support system is evaluated through specific, practical tests designed to measure whether it works correctly in real-world scenarios. Rather than applying generic worldwide metrics, the evaluation focuses on measuring the accuracy and reliability of the system components most critical to emergency operations: Does the system calculate risk scores correctly? Do retrieved incidents actually match the current scenario? Are generated precautions safe and appropriate? How fast does the system respond?

The evaluation strategy has two components. First, quantitative testing measures numerical accuracy of core system functions. Second, qualitative expert assessment validates that outputs are safe, complete, and practically useful in emergency contexts.

7.2 Component-Specific Evaluation

Each major system component has specific evaluation requirements: following are some methods that can be used to judge the model

7.2.1 1. Risk Quantification Accuracy

The risk scores (Frequency and Consequence) are the foundation of system output. Evaluating their accuracy is critical.

Consequence Score Testing

What we're testing: Does the system correctly identify severity (1-5) from incident descriptions?

Test Method: Create a set of 15-20 test incident descriptions with known correct consequence scores. For example:

- Test query containing "explosion" should produce Consequence = 5
- Test query containing "equipment failure" should produce Consequence = 2
- Test query containing "fatality" should produce Consequence = 5

Run the system on each test query, record the Consequence score it produces, compare against the expected score.

Success Metric: % Accuracy = (Number of Correct Scores / Total Test Queries) × 100

Acceptable Threshold: At least 90% of test queries produce correct Consequence scores. If accuracy is below 85%, the keyword dictionary needs revision.

Frequency Score Testing

What we're testing: Does the IEF × PFD calculation produce correct Frequency scores (1-5)?

Test Method: Create test scenarios where you control the inputs:

- Create a test query with 3 very similar incidents in the database (high similarity scores)
- Record the calculated IEF value
- Record whether PFD values were correctly identified from incident text
- Verify the final $f_C = \text{IEF} \times \text{PFD}$ calculation
- Check if f_C is correctly mapped to 1-5 scale using thresholds

Success Metric: Verify each calculation step:

- IEF calculation accuracy: Is weighted count divided by 11 years correct?
- PFD identification: Were protection layers correctly detected in incident text?
- Multiplication: Is $f_C = \text{IEF} \times \text{PFD}$ mathematically correct?
- Scaling: Is f_C value correctly mapped to 1-5 scale using thresholds?

Risk Matrix Mapping Testing

What we're testing: Given Frequency and Consequence scores, does the system correctly produce the Risk Category?

Test Method: Test all 25 possible combinations of Frequency (1-5) × Consequence (1-5):

- Manually set Frequency = 3, Consequence = 5 → should output HIGH
- Manually set Frequency = 1, Consequence = 1 → should output LOW
- Manually set Frequency = 5, Consequence = 5 → should output CATASTROPHIC
- Verify all 25 combinations against the risk matrix

Success Metric: 100% of Risk Matrix lookups should be correct. If any are wrong, it indicates a bug in the matrix implementation.

7.2.2 2. Retrieval Quality Evaluation

The system retrieves the top-3 most similar incidents. Do these actually match the user's scenario?

Similarity Verification

What we're testing: Are retrieved incidents semantically similar to the user query?

Test Method: For each test query, examine the retrieved incidents:

- Are they about the same chemical type? (e.g., chlorine leak query should retrieve chlorine incidents)
- Do they describe similar hazards? (e.g., pressure relief failure should match relief valve problems)
- Are the incident scenarios applicable to the current situation?

Success Metric: Have domain experts rate retrieved incidents:

- Rank 1 incident: Should be 80%+ similar (very relevant)
- Rank 2 incident: Should be 70%+ similar (relevant)
- Rank 3 incident: Should be 60%+ similar (somewhat relevant)

Acceptable Threshold: At least 80% of test queries should have all 3 retrieved incidents rated as relevant or very relevant. If relevance is lower, the similarity search may need tuning or the knowledge base may have limited coverage.

7.2.3 3. Precaution Generation Evaluation

The system generates precautions through two models. Evaluate both:

Dataset-Based Precautions

What we're testing: Are extracted recommendations from retrieved incidents actually appropriate for the current scenario?

Test Method:

1. For each test query, examine the 3 recommendations extracted from retrieved incidents
2. Ask a domain expert: "Would you recommend these actions in this scenario?"
3. Expert rates each precaution as: Safe/Appropriate, Neutral, or Unsafe

Success Metric: Percentage of precautions rated "Safe/Appropriate" by experts

- Target: $\geq 90\% appropriateness$
- Acceptable: 75-90%
- Poor: <75% (indicates knowledge base recommendations need review)

AI-Generated Precautions

What we're testing: Are TinyLLaMA-generated precautions safe, practical, and grounded in the retrieved incidents?

Test Method:

1. Run system on test query
2. Examine the 5 AI-generated precautions
3. Expert evaluates each precaution across 3 dimensions:
    - **Safety:** Is this precaution safe? Could it cause harm if followed?
    - **Appropriateness:** Does it match the scenario and incident type?
    - **Grounding:** Is it supported by the retrieved incidents or is it hallucinated?

Success Metric: Create a scoring rubric:

- Safe, Appropriate, Grounded = 3 points
- Safe, Appropriate, Partially Grounded = 2 points
- Safe but marginal appropriateness = 1 point
- Unsafe or clearly hallucinated = 0 points

Average score across all precautions:

- >2.5/3: Excellent (can deploy with confidence)
- 2.0-2.5/3: Good (acceptable with human oversight)
- 1.5-2.0/3: Marginal (needs improvement)
- <1.5/3: Poor (requires significant redesign)

7.2.4 4. Response Latency

What we're testing: Is the system fast enough for emergency operations?

Test Method: Measure response time for 30 representative test queries:

- Record timestamp when query is submitted
- Record timestamp when system returns complete response
- Calculate: Response Time = Completion Time - Submission Time

Success Metric: Average response time and percentile latencies:

- Target: Average response <3 seconds, 95th percentile <5 seconds
- Acceptable: Average <5 seconds, 95th percentile <10 seconds
- Unacceptable: Average >10 seconds (too slow for emergency response)

Test Component	Test Cases	Success Criteria
Consequence Score Accuracy	15-20	>90% correct
Frequency Score Calculation	10	100% correct math
Risk Matrix Mapping	25	100% correct lookups
Retrieval Similarity	20	>80% relevant
Dataset Precautions Safety	20	>90% appropriate
AI-Generated Precautions	20	Average score >2.0/3
Response Latency	30	Average <5 sec

Table 7.1: Testing Plan: Components, Test Cases, and Success Criteria

7.3 Test Execution Plan

To validate the system before operational deployment, follow this structured testing approach:

7.4 Expert Review Focus Areas

Beyond automated testing, domain experts evaluate system outputs for real-world appropriateness. Experts assess five critical dimensions:

1. **Correctness Against Safety Practices:** Do recommendations align with established safety procedures? Would a professional safety engineer approve them?
2. **Completeness of Response:** Does the system address all critical aspects of the scenario? Are there missing considerations that could affect safety?
3. **Clarity and Actionability:** Can an operator understand the recommendations and immediately translate them into specific actions? Is the guidance clear enough for high-stress situations?
4. **Appropriate Confidence Expression:** Does the system avoid overconfidence? Does it indicate when expert verification is needed? Are uncertainty statements accurate?
5. **Safety and Contraindications:** Are there any recommendations that could be dangerous? Are there inappropriate oversimplifications that ignore important details?

Expert reviewers include process safety engineers, emergency response coordinators, and facility operators who can provide practical feedback on whether the system would actually help or hinder emergency decision-making.

7.5 Handling Test Results and Iterations

For each failure case, conduct root cause analysis:

- Was it a keyword dictionary issue? (Update keywords)
- Was it a retrieval failure? (Improve similarity search or knowledge base)

- Was it an AI generation error? (Adjust prompt or confidence thresholds)
- Was it a calculation bug? (Fix implementation)

Iterations continue until acceptable performance thresholds are reached.

Chapter 8

Results and Analysis

8.1 Component-Level Strengths and Weaknesses

The LLM-based decision support system was tested on representative emergency scenarios drawn from the incident database and synthetic cases. Testing focused on understanding how the prototype behaves in realistic situations and identifying strengths, weaknesses, and error sources. The system was exercised across four use cases: safety procedure question-answering, incident scenario analysis, regulatory guidance, and incident report summarization.

Table 8.1 summarizes the observed behavior of key system components based on manual testing and expert review.

Component	**Observed Strengths**	**Typical Limitations / Failure Modes**
Retrieval (FAISS + embeddings)	Consistently retrieves semantically similar incidents when query mentions chemical, equipment and failure mode clearly	Struggles when query is vague, uses uncommon synonyms, or describes scenarios not well represented in database
Consequence scoring (keyword-based)	Quickly identifies high-severity scenarios (explosions, fatalities) with conservative scoring	Sensitive to wording; may under-score if severe terms absent, or over-score if strong words appear in non-critical context
Frequency scoring (IEF × PFD)	Transparent, formula-based calculation grounded in LOPA concepts	Dependent on small sample of similar incidents; results noisy for rare events or sparse data
Dataset-based precautions (Model 1)	When recommendations present in source reports, they are realistic and grounded in real incidents	Many incident files lack recommendation text; output can be empty or too brief
AI-generated precautions (Model 2)	Able to synthesize clear, structured precaution lists and adapt to current query	Quality depends on retrieved context; can become generic if relevant details missing
User interface and latency	Typical end-to-end response 2-5 seconds; clear presentation of scores, similar cases, precautions	Risk scores may appear more precise than justified; users may over-trust numerical outputs

Table 8.1: Qualitative Behavior of Major System Components

The system performs best when queries clearly specify chemical, equipment, and abnormal condition, similar incidents with good descriptions exist in the database, and the required answer can be constructed directly from retrieved cases. Performance degrades when queries lack critical context, database contains only loosely related incidents, or scenarios involve unusual combinations not well represented in the dataset.

8.2 Sources of Error and Limitations

Manual error analysis identifies specific sources of error particularly relevant for this prototype. Understanding these limitations is essential for appropriate deployment and indicates priority areas for future improvement.

8.2.1 Keyword-Based Consequence Scoring

The current Consequence score relies on a simple keyword dictionary mapping. This design choice prioritizes simplicity and interpretability but introduces predictable limitations:

Observed limitations:

- **Sensitivity to wording:** If a description uses softer language ("overpressure" instead of "explosion"), the system may assign a lower severity than an expert would. A single synonym variation can change the score from 5 to 3.
- **Loss of contextual nuance:** The same word appears in both major and minor contexts ("fire drill" versus "major reactor fire"), but the model treats them identically. Context that a human safety expert would immediately recognize is lost.
- **Single-trigger behavior:** The presence of one severe keyword forces the maximum score, even if the overall incident description suggests a less severe outcome. For example, "small controlled fire in well-isolated equipment" containing the word "fire" immediately triggers Consequence = 4, though the context suggests Consequence = 2 or 3.

Why this matters: Consequence scores directly feed into the Risk Matrix. Over-scoring leads to overstated risk and unnecessary escalation. Under-scoring (though less likely with the current conservative approach) could miss genuinely hazardous situations.

Potential improvements:

- Replace pure keyword lookup with a small supervised classification model that uses the full incident description (or embedding) to predict severity. Train on expert-labeled incidents from the database.
- Incorporate multiple signals beyond text keywords: chemical toxicity data, inventory quantities, occupancy numbers, proximity to residential areas.
- Use phrase-level patterns rather than single keywords. For example, distinguish "controlled small fire" (lower severity) from "uncontrolled fire" (higher severity) through pattern matching.

8.2.2 Frequency Estimation and Data Sparsity

The Frequency score depends on the number and recency of similar incidents in the knowledge base. With limited data, several issues emerge:

Observed limitations:

- **Statistically weak estimates:** Some incident types have only one or two examples in the database. Calculating IEF from such small samples produces unreliable estimates. A single new chlorine incident can dramatically change the calculated frequency.
- **Reporting bias:** Publicly available incident data reflects reporting patterns, not true accident frequencies. Some regions, operators, or incident types may be over- or under-represented.
- **Missing protection layer information:** Determining PFD values depends on identifying protection layers (SIS, relief valves, alarms) in incident text. When this information is absent or unclear, the system must fall back on severity-based PFD estimation, which is coarse.

Why this matters: A calculated Frequency = 2 (Rare) might be based on a single incident occurrence, making it no more reliable than random. Users may trust the number without understanding its true uncertainty.

Potential improvements:

- Expand the database with additional public incident sources (international databases, industry standards organizations, detailed LOPA studies).
- Explicitly encode uncertainty bands for f_C rather than presenting a single number. For example, display "Frequency = 2 (Rare), estimated from 2 similar incidents, confidence uncertain" instead of just "2/5".
- Allow expert-specified baseline frequencies for certain equipment types and failure modes, which can be combined with or constrain data-driven estimates.

8.2.3 Missing Recommendations and Dual-Model Precaution Strategy

A critical limitation of the underlying incident dataset is that many investigation reports do not contain a populated `recommendations` field. This practical constraint motivated the two-model precaution design:

Model 1 (Dataset-based precautions): When recommendations are present in similar incidents, they are extracted and displayed. These recommendations are strongly grounded in real incident investigations but often absent from the data.

Model 2 (AI-generated precautions): When Dataset-based recommendations are missing or sparse, TinyLLaMA generates precautions using retrieved incident descriptions and the user query context.

Why this dual approach was necessary: Many incident investigation reports focus on root cause analysis without explicitly listing preventive actions. Relying solely on extracted recommendations would frequently return empty results, making the system unhelpful. By incorporating AI generation, the system always returns some guidance.

Observed limitations of this approach:

- **Lack of ground truth for AI-generated cases:** When recommendations come from AI generation, there is no authoritative source to validate against. A precaution might sound reasonable but be incomplete or inappropriate for the specific context.

- **Over-generalization when data is sparse:** If retrieved incidents are only loosely related, the AI model may generalize too much, producing generic advice that does not address specific facility characteristics or chemical properties.

- **User confusion from discrepancies:** When the same scenario produces different precautions from Model 1 versus Model 2, it may confuse users about which to trust or whether the system is inconsistent.

Why this matters: This is a fundamental design trade-off. Without AI generation, the system often has no recommendations to offer. With AI generation, recommendations are always available but may lack grounding.

Potential improvements:

- Clearly label in the UI whether each precaution is "Dataset-based (from real incident investigation)" or "AI-generated (from similar cases and expert reasoning)". Use different visual styling or badges.

- When recommendations are absent in source reports, prompt the AI to generate "Candidate recommendations requiring expert review" with explicit caveats rather than treating AI output as equivalent to documented best practices.

- Implement a post-processing comparison step: when both Model 1 and Model 2 produce precautions for the same scenario, flag large discrepancies and ask the model to explain differences or consolidate recommendations.

- Prioritize dataset enrichment: systematically add recommendation text to incident reports that currently lack it, reducing reliance on AI generation over time.

8.2.4 Retrieval Limitations and Coverage Gaps

Although semantic search through embeddings provides substantial improvement over keyword matching, retrieval failures remain:

Observed limitations:

- **No close match scenarios:** Some novel scenarios have no truly similar incident in the database. The "best" retrieved cases may still be weakly related, leading to poor-quality context for precaution generation.

- **Vocabulary sensitivity:** Slight variations in wording or chemical nomenclature (common names versus IUPAC names, trade names) may reduce similarity scores even for essentially identical hazard scenarios.

- **Missing facility context:** Facility-specific details (building layout, local procedures, operator training levels) are not fully captured in incident descriptions, limiting how well retrieved cases match the actual context.

Why this matters: Poor retrieval leads to irrelevant context being passed to the AI model, which then generates precautions that may miss facility-specific considerations.

Potential improvements:

- Enrich embedding text with structured metadata tags (chemical family, equipment category, consequence class, country). This enables hybrid retrieval combining semantic similarity with structured filtering.
- Implement a minimum similarity threshold below which the system explicitly warns the user: "No strongly similar past incidents found in database. Precautions below are generic and require expert verification."
- Allow operators to manually filter or re-rank retrieved cases by chemical type, facility type, or country to improve contextual fit.

8.2.5 User Perception and Over-Confidence in Numerical Scores

A subtle but important source of error is the user's perception of system reliability based on visual presentation:

Observed limitations:

- **False precision:** Displaying "Frequency: 2/5" creates an impression of precision (the system clearly calculated this). In reality, the estimate may be based on only one or two similar incidents, making it statistically weak.
- **Risk category salience:** Users focus heavily on the Risk Matrix color (LOW=green, HIGH=red, CATASTROPHIC=dark red) and may skim explanatory text that describes underlying uncertainty.
- **Implicit trust:** A system that provides coherent-looking numbers with professional formatting builds confidence, even when the underlying data are limited.

Why this matters: Over-confidence in system outputs undermines appropriate human oversight. Users might escalate or de-escalate response based on scores that don't actually warrant that level of confidence.

Potential improvements:

- Add qualitative labels alongside scores, such as "Frequency = 2, based on only 1 similar incident — confidence LOW".
- Display brief explanations: "This score reflects X similar incidents in an 11-year database. Limited data means this estimate has considerable uncertainty."
- Require explicit user acknowledgment (e.g., a checkbox) of limitations before viewing detailed recommendations, reinforcing that scores support but do not replace expert judgment.

8.3 Key Innovation: Dual-Model Precaution Generation

One significant contribution of this work is the recognition and design solution for the missing-recommendation problem. Unlike most LLM-based systems that rely entirely on AI generation, this system explicitly incorporates both dataset-grounded and AI-generated precautions:

- **Transparency:** Users see which precautions come from documented incidents versus AI reasoning, enabling more informed judgment.
- **Grounding:** Dataset-based precautions provide a proven reference point. When AI recommendations differ, users can ask why.
- **Practical coverage:** By always offering recommendations (either from data or AI), the system remains useful even when similar incidents lack explicit guidance documents.

This dual-model approach is not a perfect solution to the missing-data problem, but it represents a pragmatic balance between completeness and trustworthiness.

Chapter 9

Conclusion and Future Work

9.1 Summary of System Development

This project developed and evaluated a prototype LLM-based decision support system for chemical emergency operations. The system integrates TinyLLaMA with a curated knowledge base of incident case studies, employing LOPA-based risk quantification and dual-model precaution generation. Through retrieval-augmented generation, the system grounds AI outputs in verified incident data, reducing hallucination risk. The evaluation identified specific strengths in common scenarios and important limitations for novel situations.

The core innovation is the recognition that existing incident investigation data lacks explicit recommendations, motivating a dual-model approach: extracting proven precautions from similar incidents while also generating adaptive recommendations through AI. This pragmatic design balances completeness with grounding.

9.2 System Limitations and Appropriate Boundaries

The system operates within important limitations that must be clearly understood by users and decision-makers considering deployment.

9.2.1 Knowledge Base and Coverage

The knowledge base contains incidents from 2010-2024 and is comprehensive but finite. Recent regulatory changes, new chemical hazards, or incidents occurring after the last database update are not reflected. The system cannot cover all possible emergency scenarios globally, and regulatory requirements vary significantly by country, region, and facility type.

System performance degrades predictably for scenarios significantly different from those in the knowledge base. Novel combinations of factors not previously encountered will receive weaker support. The appropriate scope for deployment is emergency decision-making in contexts similar to those represented in the knowledge base.

9.2.2 Real-Time Limitations

The system cannot autonomously detect emergencies. It operates on text descriptions of scenarios provided by users. It cannot directly access or sense plant conditions, cannot

verify that described conditions match actual plant state, and cannot incorporate real-time sensor data for dynamic situational awareness.

Integration with real-time monitoring systems (discussed in Section 9.4.4) could address this limitation, but the current prototype requires human description of the emergency state.

9.2.3 Reasoning and Decision Complexity

The system may struggle with scenarios requiring complex multi-factor reasoning, unusual combinations of competing considerations, or decisions in genuinely novel contexts. It is designed for trained operators and emergency managers but does not replace expert human judgment, particularly for novel or severe scenarios.

9.2.4 Communication and Visualization

The system communicates via text only. It cannot see facility layouts, equipment drawings, process flow diagrams, or visual information that might be important for emergency decisions. Integration with facility visualization systems could enhance this capability in future versions.

9.3 Enhancement Opportunities for Future Development

While the prototype demonstrates proof-of-concept capabilities, several enhancements could significantly increase effectiveness and readiness for industrial deployment.

9.3.1 User Experience: Trend Analysis and Safeguard Visualization

Current system operation is reactive—it analyzes each query independently. A valuable enhancement would implement proactive trend visualization showing whether particular hazard patterns are becoming more or less frequent over time. A line graph displaying incident counts from similar cases grouped by year (for example, last 10 years) would help users identify emerging or declining risk patterns.

A second enhancement would visualize control layer gaps through bar charts showing which safeguards are commonly present versus frequently absent or failed in similar incidents. The system could scan retrieved cases for keywords related to SIS, relief valves, alarms, and process controllers, calculating frequency of each safeguard type. This visualization would highlight common weakness patterns, directing preventive attention toward frequent failure modes.

These enhancements transform the system from reactive analysis to proactive risk identification.

9.3.2 Predictive Capability: Mitigation Effectiveness and Root Cause Distribution

Beyond current risk scoring, the system could estimate mitigation effectiveness. Given the proposed precautions, how much would the calculated risk decrease? The LLM could be prompted to analyze its own recommendations and estimate consequence reduction (for example, "If these 5 precautions were implemented, would Consequence drop from 5/5 to 3/5?").

A second enhancement would analyze root cause patterns. Instead of displaying raw cause text from incidents, the system could categorize causes from retrieved cases into buckets (Human Error, Mechanical Failure, Design Deficiency, Procedural Flaw, Environmental Factor), displaying the distribution as a chart. This helps users understand which failure modes are most common in similar incidents.

9.3.3 Data Quality: Temporal Weighting

Currently, risk calculation treats all historical incidents equally based on similarity alone. However, a recent incident (2024) provides better evidence of current risk than an identical incident from 20 years ago. Implementing temporal weighting would adjust similarity scores based on incident age:

$$\text{Weighted Similarity} = \text{Base Similarity} \times \left(1 + \frac{\text{Years Since Incident}}{\text{Max Age in Database}}\right)$$

This ensures that highly similar recent incidents dramatically increase calculated frequency compared to identical incidents from the distant past, better reflecting reality that recent patterns predict current risk.

9.3.4 Real-Time Integration

A major enhancement would integrate the system with plant monitoring systems. Rather than operating on text descriptions, the system could access real-time process data, equipment status, and environmental monitoring. This enables automated hazard detection, real-time situation assessment, and dynamic recommendation updating as conditions evolve.

Implementation requires secure APIs to plant systems, real-time data processing, and dynamic prompt construction. This represents significant architectural enhancement, moving from static scenario analysis to continuous decision support.

9.3.5 Knowledge Base Expansion and Specialization

Expanding the knowledge base would improve coverage and relevance. Comprehensive industry incident database integration would include documented incidents from chemical manufacturing, refining, pharmaceutical production, and specialty chemicals. Facility-specific procedure incorporation would allow encoding emergency procedures for particular facility types or configurations. Regulatory database integration would ensure comprehensive coverage of applicable standards. Expert knowledge capture would incorporate tacit knowledge from experienced safety practitioners.

9.3.6 Formal Safety Certification

Advancing from prototype to operational system requires formal safety processes. Safety Integrity Level (SIL) assessment would evaluate whether the system meets SIL 1 or SIL 2 requirements appropriate to decision support applications. Formal verification methods would establish mathematical guarantees about system behavior in critical scenarios. Independent safety review by external experts would provide unbiased assessment of safety and reliability. Regulatory compliance review would ensure alignment with applicable regulatory frameworks in target jurisdictions.

9.4 Conclusion

This project demonstrates that LLM-based decision support for chemical emergency management is technically feasible and can provide valuable assistance in routine, well-defined scenarios. The system successfully synthesizes information from historical incidents, calculates risk using established LOPA methodology, and generates both evidence-based and adaptive precautions.

However, the evaluation also revealed important limitations: reliance on keyword-based consequence scoring, frequency estimation with sparse data, dependence on AI generation when recommendations are missing, and challenges with novel scenarios. These limitations define the appropriate scope for current deployment and motivate enhancement directions.

The system is best viewed as a decision support tool that augments human expertise, not replaces it. With appropriate human oversight, clear communication of limitations, and ethical deployment practices, systems of this type can help emergency managers make better-informed decisions under time pressure. Future work addressing the enhancement opportunities outlined in Section 9.4 would move the system toward industry-ready capability.

The dual-model precaution approach pioneered here—combining dataset-grounded and AI-generated recommendations—addresses a fundamental challenge in applying LLMs to specialized domains where authoritative data may be incomplete. This design principle may prove valuable in other safety-critical domains beyond chemical emergency management.

Chapter 10

Appendix: Implementation Code

10.1 Model 1: Dataset-Based Precaution Extraction

This section contains the complete code for Model 1, which extracts recommendations directly from the historical incident database.

10.1.1 Libraries and Setup

```
# === MODEL 1: DATASET-BASED PRECAUTION EXTRACTION ===

# Import essential libraries
import pandas as pd
import numpy as np
import faiss
import requests
import gradio as gr
import logging
import os
import matplotlib.pyplot as plt
from sentence_transformers import SentenceTransformer

# Setup logging for debugging and monitoring
logging.basicConfig(level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s')

# Path to incident database (CSV file)
csv_path = r"C:\\Users\\Udayanraje Patil\\Downloads\\LLM_Model_FInal_data.csv"
```

10.1.2 Data Loading and Embedding Creation

```
# Load and prepare the incident database
try:
    if not os.path.exists(csv_path):
        raise FileNotFoundError(f"File not found: {csv_path}")

    # Read CSV with encoding fixes for special characters
```

```
df = pd.read_csv(
    csv_path,
    quotechar='"',
    engine='python',
    encoding='latin1'
).fillna("")

# Fix encoding artifacts in text columns
for col in df.select_dtypes(include=['object']).columns:
    df[col] = (
        df[col]
        .astype(str)
        .str.encode('latin1', errors='ignore')
        .str.decode('utf-8', errors='ignore')
    )

# Standardize column names (strip whitespace)
df.columns = [c.strip() for c in df.columns]

# Rename columns to standard names for processing
df.rename(columns={
    'File Name': 'title',
    'Cause': 'cause',
    'Direct Link': 'link',
    'Country': 'country',
    'Date': 'date'
}, inplace=True)

# Generate Case IDs if not present
if 'case_id' not in df.columns:
    df['case_id'] = df.index + 1

# Load pre-trained sentence transformer model
model = SentenceTransformer("all-MiniLM-L6-v2")

# Create search text by combining title and cause
search_text = df['title'].astype(str) + " " + df['cause'].astype(str)

# Generate semantic embeddings (384-dimensional vectors)
embeds = model.encode(search_text.tolist(),
                      show_progress_bar=True,
                      convert_to_numpy=True).astype("float32")

# Create FAISS index for fast similarity search
index = faiss.IndexFlatL2(embeds.shape[1])
index.add(embeds)

logging.info("System Loaded Successfully!")
```

```
except Exception as e:
    logging.critical(f"Setup Error: {e}")
    df = pd.DataFrame()
    index = None
```

10.1.3 Risk Calculation Logic

```
# Define LOPA Risk Matrix (Consequence x Frequency)
LOPA_RISK_MATRIX = {
    1: {1: "LOW", 2: "LOW", 3: "LOW", 4: "MEDIUM", 5: "MEDIUM"},
    2: {1: "LOW", 2: "LOW", 3: "MEDIUM", 4: "MEDIUM", 5: "SERIOUS"},
    3: {1: "LOW", 2: "MEDIUM", 3: "MEDIUM", 4: "SERIOUS", 5: "HIGH"},
    4: {1: "MEDIUM", 2: "MEDIUM", 3: "SERIOUS", 4: "HIGH", 5: "HIGH"},
    5: {1: "MEDIUM", 2: "SERIOUS", 3: "HIGH", 4: "HIGH", 5: "CATASTROPHIC"},
}

# Consequence keywords mapped to severity scores (1-5)
CONSEQ_KEYWORDS = {
    "fatal": 5, "death": 5, "explosion": 5, "rupture": 5, "catastrophic": 5,
    "fire": 4, "blast": 4, "release": 4, "vapor": 4,
    "leak": 3, "spill": 3, "injury": 3, "shutdown": 3,
    "failure": 2, "deviation": 2, "trip": 2,
    "alarm": 1, "alert": 1, "warning": 1
}

# Protection layer keywords mapped to PFD values
LAYER_KEYWORDS = {
    "sis": (["sil", "esd", "trip", "interlock"], 0.01),
    "relief": (["psv", "prv", "relief", "rupture disk"], 0.01),
    "mitigation": (["firewater", "sprinkler", "monitor"], 0.5)
}

# Extract consequence score from incident description
def get_consequence_score(text):
    """Identify highest consequence level from keywords"""
    text = str(text).lower()
    scores = [score for kw, score in CONSEQ_KEYWORDS.items() if kw in text]
    return max(scores) if scores else 1

# Calculate final risk category using LOPA matrix
def calculate_risk(query, similar_df, distances):
    try:
        # Frequency: based on similarity scores
        sim_scores = np.clip(1.0 - (distances / 2.0), 0.0, 1.0)
        frequency_val = np.mean(sim_scores) if len(sim_scores) > 0 else 0

        if frequency_val > 0.7: freq = 5
```

```
        elif frequency_val > 0.5: freq = 4
        elif frequency_val > 0.3: freq = 3
        elif frequency_val > 0.1: freq = 2
        else: freq = 1

        # PFD adjustment: reduce frequency if protection layers present
        combined_past = " ".join(similar_df['cause'].astype(str))
        if any(any(k in combined_past.lower() for k in kws)
               for sys, (kws, val) in LAYER_KEYWORDS.items()):
            freq = max(1, freq - 1)

        # Get consequence score
        cons = get_consequence_score(query)

        # Look up risk category in matrix
        cat = LOPA_RISK_MATRIX[cons][freq]
        return cat, cons, freq
    except:
        return "LOW (Default)", 1, 1

# Call local LLM for AI-generated analysis
def llm_call(prompt):
    """Send prompt to local TinyLLaMA model via Ollama"""
    try:
        r = requests.post("http://localhost:11434/api/generate",
            json={"model":"tinyllama","prompt":prompt,"stream":False},
            timeout=30)
        if r.status_code == 200:
            return r.json().get("response", "").strip()
    except:
        return "AI Analysis Unavailable."
```

10.1.4 Query Processing (Model 1)

```
# CORE PROCESSING ENGINE FOR MODEL 1
def process_query_model1(query):
    """
    MODEL 1: Extract recommendations from historical database.
    Returns: [risk_category, consequence/5, frequency/5, risk_plot,
              recommendations, case_headers, case_causes]
    """
    empty_result = ["N/A", "0/5", "0/5", None, "", "", "", "", "", "", ""]

    if df.empty or index is None:
        return ["Error: Data not loaded"] + empty_result[1:]

    # 1. RETRIEVAL: Find top-3 most similar incidents
    q_vec = model.encode([query]).astype("float32")
```

```
distances, indices = index.search(q_vec, k=3)
valid_idx = [i for i in indices[0] if i < len(df)]
top_cases = df.iloc[valid_idx]

# 2. RISK CALCULATION
risk_cat, cons_score, freq_score = calculate_risk(
    query, top_cases, distances[0])

# 3. PRECAUTION EXTRACTION (MODEL 1 ONLY)
# Priority: Use database recommendations from similar cases
advice_text = ""
source_label = ""

for i, row in top_cases.iterrows():
    rec_raw = str(row.get('recommendations', '')).strip()
    # Clean empty placeholders
    if (len(rec_raw) > 10 and
        "none" not in rec_raw.lower() and
        "not specified" not in rec_raw.lower()):
        advice_text = rec_raw
        source_label = f"**Source: Historical Data (Case #{row.get('case_id')})**
        break

# Fallback to LLM if no database recommendation found
if not advice_text:
    source_label = "**Source: AI Analysis (General Guidelines)**"
    prompt = (f"Situation: '{query}'. Provide 5 critical safety "
              "precautions for this scenario.")
    advice_text = llm_call(prompt)

final_advice = f"{source_label}\n\n{advice_text}"

# 4. GENERATE RISK VISUALIZATION
plot = generate_risk_plot(cons_score, freq_score)

# 5. FORMAT SIMILAR CASES FOR DISPLAY
case_outputs = []
for i in range(3):
    if i < len(top_cases):
        row = top_cases.iloc[i]
        cid = row.get('case_id', 'N/A')
        link = row.get('link', '#')
        date_str = str(row.get('date', '')).strip()
        if 'struct' in date_str.lower() or len(date_str) < 4:
            date_str = "Not Found"

        header = (f"**Case #{cid}** |  {date_str} | "
                  f"[ Open Report]({link})")
```

```
        raw_cause = str(row.get('cause', '')).strip()
        cause_text = (raw_cause if len(raw_cause) > 5
                      else "No specific cause detailed in report.")
    else:
        header = "**No Case Found**"
        cause_text = ""

    case_outputs.append(header)
    case_outputs.append(cause_text)

return ([risk_cat, f"{cons_score}/5", f"{freq_score}/5",
         plot, final_advice] + case_outputs)
```

10.2 Model 2: AI-Generated Precaution Synthesis

This section contains the complete code for Model 2, which uses TinyLLaMA to generate context-specific precautions.

10.2.1 Risk Calculation and Plotting

```
# === MODEL 2: AI-GENERATED PRECAUTION SYNTHESIS ===

# Define LOPA Risk Matrix (same as Model 1)
LOPA_RISK_MATRIX = {
    1: {1: "LOW", 2: "LOW", 3: "LOW", 4: "MEDIUM", 5: "MEDIUM"},
    2: {1: "LOW", 2: "LOW", 3: "MEDIUM", 4: "MEDIUM", 5: "SERIOUS"},
    3: {1: "LOW", 2: "MEDIUM", 3: "MEDIUM", 4: "SERIOUS", 5: "HIGH"},
    4: {1: "MEDIUM", 2: "MEDIUM", 3: "SERIOUS", 4: "HIGH", 5: "HIGH"},
    5: {1: "MEDIUM", 2: "SERIOUS", 3: "HIGH", 4: "HIGH", 5: "CATASTROPHIC"},
}

# Extended keyword dictionary for consequence scoring
CONSEQ_KEYWORDS = {
    "fatal": 5, "death": 5, "explosion": 5, "rupture": 5, "catastrophic": 5,
    "fire": 4, "blast": 4, "release": 4, "vapor": 4, "cloud": 4,
    "leak": 3, "spill": 3, "injury": 3, "burn": 3, "damage": 3, "shutdown": 3,
    "failure": 2, "deviation": 2, "trip": 2, "stuck": 2, "corrosion": 2,
    "alarm": 1, "alert": 1, "warning": 1, "maintenance": 1, "noise": 1
}

# Extended protection layer keywords
LAYER_KEYWORDS = {
    "sis": (["sil", "esd", "trip", "interlock", "safety instrumented"], 0.01),
    "relief": (["psv", "prv", "relief", "rupture disk", "safety valve"], 0.01),
    "alarm": (["alarm", "operator", "hmi"], 0.1),
    "control": (["controller", "control loop", "bpcs", "plc"], 0.1),
    "mitigation": (["firewater", "sprinkler", "monitor", "deluge"], 0.5)
}

# Generate visual Risk Matrix heatmap
def generate_risk_plot(cons_score, freq_score):
    """Generates a visual Risk Matrix showing current risk position"""
    fig, ax = plt.subplots(figsize=(5, 4))

    # Color matrix: Green (LOW) -> Yellow -> Red (CATASTROPHIC)
    colors = np.array([
        [0, 0, 0, 1, 1],
        [0, 0, 1, 1, 2],
        [0, 1, 1, 2, 3],
        [1, 1, 2, 3, 3],
        [1, 2, 3, 3, 4]
```

```
    ])
    cmap = plt.cm.colors.ListedColormap(
        ['#66ff66', '#ccff66', '#ffff66', '#ffcc00', '#ff3333'])

    ax.imshow(colors, cmap=cmap, origin='lower',
              extent=[0.5, 5.5, 0.5, 5.5])

    # Add grid and labels
    ax.set_xticks(np.arange(1.5, 5.5, 1))
    ax.set_yticks(np.arange(1.5, 5.5, 1))
    ax.grid(color='black', linestyle='-', linewidth=0.5)

    # Plot current risk position as black X
    ax.scatter(freq_score, cons_score, color='black', s=200,
               marker='X', edgecolors='white', linewidth=1.5)

    ax.set_xlabel('Frequency (1-5)')
    ax.set_ylabel('Consequence (1-5)')
    ax.set_title(f'Risk Position ({freq_score}, {cons_score})')

    return fig

# Calculate risk with protection layer credit
def calculate_risk(query, similar_df, distances):
    """
    Calculate risk using:
    1. Frequency based on similarity to past incidents
    2. PFD credit if protection layers found
    3. Consequence based on keywords
    4. Final risk category from matrix
    """
    try:
        # Frequency: convert similarity distances to frequency rating
        sim_scores = np.clip(1.0 - (distances / 2.0), 0.0, 1.0)
        frequency_val = np.mean(sim_scores) if len(sim_scores) > 0 else 0

        if frequency_val > 0.7: freq_rating = 5
        elif frequency_val > 0.5: freq_rating = 4
        elif frequency_val > 0.3: freq_rating = 3
        elif frequency_val > 0.1: freq_rating = 2
        else: freq_rating = 1

        # PFD Credit: Reduce frequency if safeguards identified
        combined_past_text = " ".join(similar_df['cause'].astype(str) +
                                      " " + similar_df.get('full_text', ''))
        combined_past_text = combined_past_text.lower()

        safeguards_found = any(
```

```
            any(k in combined_past_text for k in kws)
            for sys, (kws, val) in LAYER_KEYWORDS.items())

        if safeguards_found:
            freq_rating = max(1, freq_rating - 1)

        # Consequence from query text
        cons_rating = get_consequence_score(query)

        # Look up risk category
        risk_cat = LOPA_RISK_MATRIX[cons_rating][freq_rating]

        return risk_cat, cons_rating, freq_rating
    except Exception as e:
        logging.error(f"Risk Calc Error: {e}")
        return "LOW (Default)", 1, 1

# Call TinyLLaMA with deterministic generation
def llm_call(prompt):
    """
    Send prompt to local TinyLLaMA model via Ollama.
    Uses temperature=0 for deterministic (reproducible) responses.
    """
    try:
        payload = {
            "model": "tinyllama",
            "prompt": prompt,
            "stream": False,
            "options": {"temperature": 0.0}
        }
        r = requests.post("http://localhost:11434/api/generate",
                          json=payload, timeout=30)
        if r.status_code == 200:
            return r.json().get("response", "").strip()
    except:
        return "AI Analysis Unavailable."
```

10.2.2 Query Processing (Model 2)

```
# CORE PROCESSING ENGINE FOR MODEL 2
def process_query_model2(query):
    """
    MODEL 2: Generate adaptive precautions using AI.
    Considers similar incidents as context for LLM generation.
    Returns: [risk_category, consequence/5, frequency/5, risk_plot,
              ai_precautions, case_headers, case_causes]
    """
    empty_result = ["N/A", "0/5", "0/5", None, "", "", "", "", "", "", ""]
```

```
if df.empty or index is None:
    return ["System Error: Data not loaded"] + empty_result[1:]

# 1. RETRIEVAL: Find top-3 most similar incidents
q_vec = model.encode([query]).astype("float32")
distances, indices = index.search(q_vec, k=3)
valid_idx = [i for i in indices[0] if i < len(df)]
top_cases = df.iloc[valid_idx]

# 2. RISK CALCULATION
risk_cat, cons_score, freq_score = calculate_risk(
    query, top_cases, distances[0])

# 3. BUILD CONTEXT FROM SIMILAR CASES (MODEL 2)
context = ""
case_ids = []
for i, row in top_cases.iterrows():
    context += (f"Case: {row.get('title')}\n"
                f"Cause: {row.get('cause')}\n\n")
    case_ids.append(f"#{row.get('case_id')}")

ref_string = ", ".join(case_ids)

# 4. GENERATE PRECAUTIONS USING LLM (MODEL 2)
prompt = (f"""You are a safety expert. Situation: "{query}".
Review these similar past accidents: {context}.
List 5 critical, immediate safety precautions the operator must take.""")

advice_content = llm_call(prompt)

# Final advice with case references
final_advice = (f"**References: Analysis based on similar Cases "
                f"{ref_string}**\n\n{advice_content}")

# 5. GENERATE RISK VISUALIZATION
plot = generate_risk_plot(cons_score, freq_score)

# 6. FORMAT SIMILAR CASES FOR DISPLAY
case_outputs = []
for i in range(3):
    if i < len(top_cases):
        row = top_cases.iloc[i]
        cid = row.get('case_id', 'N/A')
        title = row.get('title', 'Unknown Case')

        # Country check
        country = str(row.get('country', 'Not Given')).strip()
```

```
            if not country or country.lower() == 'nan':
                country = "Not Given"

            # Date check
            date = str(row.get('date', 'Not Given')).strip()
            if (not date or 'struct' in date.lower() or len(date) < 4):
                date = "Not Found"

            link = row.get('link', '#')

            header = (f"**{title} (Case #{cid})**\n"
                      f" {country} |  {date} | [ Open Report]({link})")

            # Cause text
            raw_cause = str(row.get('cause', '')).strip()
            if len(raw_cause) < 5:
                cause_text = "No specific cause detailed in report."
            else:
                cause_text = raw_cause
        else:
            header = "**No Case Found**"
            cause_text = ""

        case_outputs.append(header)
        case_outputs.append(cause_text)

    return ([risk_cat, f"{cons_score}/5", f"{freq_score}/5",
             plot, final_advice] + case_outputs)
```

10.3 Comparison: Model 1 vs Model 2

Aspect	**Model 1 (Dataset-Based)**	**Model 2 (AI-Generated)**
Data Source	Extracted from historical incident recommendations	Generated by LLM using similar cases as context
Reliability	High (based on proven practices)	Medium (depends on LLM quality and context)
Completeness	May be sparse if data missing	Always provides recommendations
Adaptability	Fixed to historical scenarios	Adapts to novel situations
Processing	Direct database lookup	Requires LLM API call
Use Case	Well-documented incidents	Novel or rare scenarios

Table 10.1: Comparison of Model 1 and Model 2 Precaution Generation Approaches

Bibliography

Major References

[1] Roy, S., & Kshirsagar, R. (2021). Development of risk acceptance criteria in the Indian context. *Process Safety and Environmental Protection*, 148, 358–369. Available at: https://doi.org/10.1016/j.psep.2020.10.021

[2] Center for Chemical Process Safety (CCPS). (2009). *Guidelines for Developing Quantitative Safety Risk Criteria*. John Wiley & Sons. Overview: https://www.aiche.org/ccps

[3] Center for Chemical Process Safety (CCPS). (2001). *Layer of Protection Analysis: Simplified Process Risk Assessment*. American Institute of Chemical Engineers.

[4] Aven, T. (2015). *Risk Analysis*. John Wiley & Sons.

[5] Rausand, M. (2011). *Risk Assessment: Theory, Methods, and Applications*. John Wiley & Sons.

Minor / Supporting References

[6] Lees, F. P. (2012). *Lees' Loss Prevention in the Process Industries* (4th ed.). Butterworth–Heinemann.

[7] Kletz, T. (2009). *What Went Wrong? Case Histories of Process Plant Disasters and How They Could Have Been Avoided* (5th ed.). Gulf Professional Publishing.

[8] Health and Safety Executive (HSE). (2001). *Reducing Risks, Protecting People: HSE's Decision-Making Process*. Available at: https://www.hse.gov.uk/risk/theory/r2p2.htm

[9] Aven, T., & Vinnem, J. E. (2007). *Risk Management: With Applications from the Offshore Petroleum Industry*. Springer.

This marks the end of the B.Tech Technical Project Report on

Process Safety and Risk Management Using Large Language Models

Department of Chemical Engineering
Indian Institute of Technology, Bombay

Thank you for reading.