**1. Project Overview & Objectives**

**1.1 Business Context**

Larsen & Toubro (L&T) operates in a highly complex and compliance-intensive engineering environment, where real-time insights into project execution, quality deviations, and field changes are critical. Each day, engineering and quality teams manage and analyze thousands of **Non-Conformance Reports (NCRs)** and **Field Change Documents (FCDs)**.

Traditionally, accessing insights from these documents required:

* Writing manual SQL queries or applying Excel filters.
* Spending **15–20 hours per week** per team just for data extraction.
* Multiple back-and-forth calls across teams before action meetings.

These manual processes resulted in:

* **Delayed decision-making**, especially during audits or execution bottlenecks.
* **Human errors** in reporting and non-standard interpretations of data.
* **Limited visibility** at the top-management level, as insights were buried in raw logs.

Thus, the need emerged for a **natural language-driven AI assistant** that simplifies data access, enhances visibility, and supports agile, data-backed decision-making at all levels.

**1.2 Problem Statement**

Despite having well-structured digital records, the **manual querying of NCR and FCD data** significantly hinders operational efficiency:

* Engineering managers **lose 15–20 hours/week** on non-strategic tasks like pulling counts, filtering records, and preparing reports.
* The **lack of intuitive interfaces** for non-technical users means data remains underutilized.
* During audits or project review meetings, updates are often **delayed or inconsistent**, affecting compliance and decision velocity.
* As data volumes grow, this manual system becomes **non-scalable** and error-prone.

This project aims to replace these inefficiencies with a conversational AI tool that understands engineering jargon and provides **instant, trustworthy, and context-aware insights**.

**1.3 Project Goals**

| **Objective** | **Technical Approach** | **Business Outcome** |
| --- | --- | --- |
| Automate NCR/FCD querying | **NLP-to-Code conversion via LangChain RAG** | ⏱️ 60% faster data retrieval |
| Enable natural language interface | **Streamlit-based UI with LangChain memory** | 🔧 No-code access for engineers |
| Improve data-driven decisions | **Dynamic visualization using Seaborn/Matplotlib** | 📊 Real-time project insights |
| Ensure data security & compliance | **Sandboxed execution (safe\_execute\_pandas\_code)** | 🔒 Secure on-premise deployment |

These goals align directly with business KPIs such as audit readiness, productivity enhancement, and cross-functional collaboration.

**1.4 Stakeholder Value Proposition**

This AI-powered assistant delivers tangible value to various stakeholder groups:

* **Engineering Teams:** Instantly access NCRs or FCDs using simple questions like *"Show WIP NCRs for Dolvi project"* without writing a single line of code.
* **Quality & Compliance Heads:** Get real-time metrics like *"Count overdue FCDs"* to ensure faster resolutions and avoid compliance risks.
* **Top Management:** Gain **strategic oversight** of bottlenecks, discipline-wise issues, or project progress via auto-generated plots and summaries.
* **IT & Data Teams:** Deploy a reusable, secure NLP interface that can be extended to other departments such as:
  + **Procurement** (e.g., *“Show POs pending approval”*),
  + **HR** (e.g., *“List employees overdue for training”*),
  + **Finance** (e.g., *“Show invoices over 90 days”*).

Ultimately, this tool empowers **business users** to drive **data-backed decisions** without relying on manual reports or technical teams.

**1.5 Scope and Limitations**

**✅ In-Scope (Current Capabilities):**

* Support for **five major query types**:
  + Count (e.g., *How many FCDs are open?*)
  + Table (e.g., *List all NCRs for Debari project*)
  + Plot (e.g., *Visualize NCRs by Discipline*)
  + Summary (e.g., *Describe this NCR document*)
  + Dashboard *(Planned)*
* Understands **domain-specific terms**, e.g.,:
  + "WIP" → "WORK IN PROGRESS"
  + "fcd" or "ncr" → mapped to correct DataFrame
* Supports **RAG-based prompting** to keep context of conversation.
* Handles **on-premise deployment** with secure and sandboxed code execution using safe\_execute\_pandas\_code().

**❌ Out-of-Scope / Limitations:**

* Currently handles only **structured tabular data** (e.g., .parquet format).
* No voice interface yet — **speech-to-text integration** (e.g., NVIDIA Riva or Whisper) is a **future enhancement**.
* Assumes **data schema consistency** — column renaming must be handled during pre-processing.
* **Multi-language or multilingual query support** not included in the current build.

**2. System Architecture & Design**

**2.1 Overall Architecture Diagram**

* **Description**: Illustrates the interaction flow between user inputs, AI model, data layer, and output rendering.
* **Components**:
  + User interface (Streamlit): Accepts queries and displays results.
  + LangChain logic: Interprets intent and manages memory.
  + Nvidia LLM: Converts natural language into executable Python/Pandas code.
  + FAISS vector store: Enhances semantic retrieval of document context.
  + DataFrames (df\_FCD, df\_NCR): Structured data for querying.
  + Output layer: Displays results or plots depending on query intent.

*(Include a diagram showing arrows from User → Streamlit UI → LangChain + LLM → Code Execution → Output Display/Plot)*

**2.2 Technology Stack Used**

| **Component** | **Technology** | **Role** |
| --- | --- | --- |
| UI Layer | Streamlit | Interactive chat interface |
| AI Logic | NVIDIA LLM + LangChain | Translates user intent to executable Python code |
| Data Storage | Parquet (df\_FCD, df\_NCR) | Source of structured enterprise data |
| Vector Retrieval | FAISS | Enables similarity-based search on document descriptions |
| Visualization | Seaborn/Matplotlib | Plots insights on delay, status, discipline, etc. |

**2.3 Data Sources (FCD & NCR Parquet Files)**

* **df\_FCD (Field Change Documents)**:
  + Contains change requests related to field engineering.
  + Key fields: Order\_Description, Discipline, Status, Approved\_By, Delay\_Days.
* **df\_NCR (Non-Conformance Reports)**:
  + Tracks quality issues in engineering.
  + Key fields: Project, Discipline, Status, Delay\_Category, Closed\_By.
* **Business Use**: These datasets reflect real-world bottlenecks and compliance risks in large engineering projects. Insights derived help in resource planning, escalation management, and delay mitigation.

**2.4 LangChain Workflow**

* **Intent Classification**: Uses keyword mapping to classify queries into count, table, plot, or dashboard.
* **Prompt Construction**:
  + For table and count queries: Constructs a Pandas prompt instructing the LLM to filter or summarize.
  + For plot queries: Constructs a visualization-specific prompt using Seaborn/Matplotlib instructions.
* **Memory Management**: LangChain’s ConversationBufferMemory retains the context of past queries.
* **Routing**:
  + Based on intent and keywords (FCD, NCR), selects the appropriate DataFrame and code path.

**2.5 NVIDIA LLM & Prompt Engineering**

* **Role of LLM**:
  + Translates natural language into syntactically correct, secure Python code.
  + Supports both data filtering and plotting instructions.
* **Prompt Engineering**:
  + Ensures LLM restricts output to only valid, safe Pandas/Seaborn code.
  + Avoids hallucinations by supplying schema-aware column lists.
  + Enforces Streamlit rendering with st.pyplot() and st.dataframe() for clean output.
* **Business Relevance**:
  + Provides stakeholders with instant, AI-powered visibility into project bottlenecks.
  + Saves analyst time by eliminating the need to manually write filters or visualizations.
  + Helps project managers make data-driven decisions during reviews.

**3. Technical Implementation**

**3.1 Intent Classification Logic**

At the core of the system lies a simple yet powerful rule-based intent classifier that determines the nature of the user's question. Based on keywords in the query, it routes the request to appropriate logic paths.

**3.1.1 Count**

* **Keywords matched**: how many, count, number of, total number.
* **Action**:
  + Constructs a Pandas code prompt that aggregates data using .count() or .value\_counts().
  + Returns scalar values (e.g., number of open NCRs).
* **Business Value**: Helps users get instant metrics on quality or design documentation load.

**3.1.2 Table**

* **Keywords matched**: list, display, show, filter, which, pending, entries, etc.
* **Action**:
  + Generates filtering logic in Pandas (df[df['Status'] == 'Open']) and returns as a DataFrame.
  + Streamlit st.dataframe() is used to render results.
* **Business Value**: Provides operational clarity—e.g., engineers can quickly see unresolved NCRs.

**3.1.3 Plot**

* **Keywords matched**: visualize, chart, plot, bar graph, pie chart, draw, graph.
* **Action**:
  + Constructs prompt instructing the LLM to generate Seaborn/Matplotlib code.
  + Uses sns.countplot(), sns.barplot(), or plt.pie() depending on context.
  + Ends with plt.tight\_layout() and st.pyplot() for seamless Streamlit display.
* **Business Value**: Enables project stakeholders to visually interpret trends—discipline-wise FCD volume, NCR delays, status breakdown, etc.

**3.1.4 Summary**

* **Keywords matched**: what is this document, describe, theme, insights, key issues.
* **Action**:
  + Retrieves relevant document chunks using FAISS + LangChain retriever.
  + Generates summarization using LLM (run(context)).
* **Business Value**: Summarizes long engineering reports or issue logs for non-technical managers.

**3.1.5 Dashboard**

* **Keywords matched**: overview, dashboard, panel, summary view.
* **Action**:
  + [Pluggable] Reserved for multi-metric, multi-plot rendering.
  + Could combine tables, counts, and charts in one output.
* **Business Value**: Future extension to support project-level visualization panels.

**3.2 Context Memory & Query History**

* **Tool**: ConversationBufferMemory from LangChain.
* **Purpose**:
  + Retains prior user queries and assistant responses.
  + Enables the system to maintain context during a session, such as “now plot that” referring to the last query.
* **Example**:

python

CopyEdit

memory.chat\_memory.add\_user\_message(query + " [FCD]")

memory.chat\_memory.add\_ai\_message(generated\_code)

* **Business Use**: Allows cumulative exploration of data—users can narrow, compare, or sequence questions.

**3.3 Safe Code Execution Pipeline**

* **Function**: safe\_execute\_pandas\_code(...)
* **Security Measures**:
  + Removes non-ASCII characters from code.
  + Sanitizes use of print(...) by replacing it with Streamlit display.
  + Validates syntax with ast.parse() before executing.
* **Execution Layer**:
  + Executes within a sandboxed exec() environment using local variables like df\_NCR, df\_FCD, plt, sns.
  + Handles intent == "plot" using plt.gcf() and Streamlit's st.pyplot().
  + Handles intent == "table" by searching for result DataFrames like filtered\_df, output\_df.
* **Business Relevance**:
  + Guarantees robust and secure code execution.
  + Ensures user trust by preventing unsafe or ambiguous behavior.

**3.4 Streamlit Front-End UX Features**

* **Chat Interface**:
  + Query box with placeholder suggestions (e.g., "List all FCDs in WIP").
  + Submit button with animated spinner and icons.
  + Custom chat history rendering with alternating user and assistant messages.
* **Output Layer**:
  + Displays Pandas tables using st.dataframe(...).
  + Displays plots inline using st.pyplot(...).
  + Displays raw code for debugging (optional toggle).
* **Styling**:
  + Uses custom layout with logos and column spacing (col1, col2).
  + Includes markdown headers, colored warnings, and plot subheaders based on content type.
* **Business Viewpoint**:
  + Designed for non-technical users: engineers, supervisors, and project managers.
  + Low learning curve. Familiar UI feel mimicking consumer-grade chatbots.

**3.5 Plotting Engine: Matplotlib & Seaborn**

* **Why Seaborn + Matplotlib**:
  + Seaborn handles high-level statistical plots easily (sns.barplot, sns.countplot, sns.histplot).
  + Matplotlib complements with layout control and rendering.
* **Usage in Code**:
  + Code ends with plt.tight\_layout(); plt.show() which gets rewritten to st.pyplot(plt.gcf()) for Streamlit.
  + Supports multiple chart types: bar, pie, line, count distribution.
  + Default plot size customized via plt.figure(figsize=(7,4)) for compact display.
* **Business Benefit**:
  + Enables at-a-glance decision-making.
  + Converts raw operational data into visuals useful for weekly project reviews or dashboards.

**4. Python Code Snippets (Critical Parts)**

This section presents and explains the most essential parts of the system's Python code. These components form the backbone of the AI-powered NCR-FCD Insight Engine, handling everything from user intent detection to dynamic visualization rendering and safe code execution.

**4.1 safe\_execute\_pandas\_code()**

**Purpose**:  
This function securely executes Python (Pandas + plotting) code generated by the LLM, ensuring that any logic (table, count, or plot) is safely evaluated and correctly displayed via Streamlit.

**Key Features**:

* **Input validation**: Ensures code is a valid string and safely parsable using ast.parse().
* **Code sanitation**: Cleans non-ASCII characters and modifies plt.show() → st.pyplot(...) for Streamlit compatibility.
* **Safe execution**: Executes in a controlled exec() environment with pre-defined variables (df\_NCR, df\_FCD, plt, sns, etc.).
* **Intent handling**:
  + If intent == "plot": Displays plot with size check and auto-clear via plt.clf().
  + If intent == "count": Returns scalar result from print(...) output.
  + Else: Detects DataFrame (filtered\_df, output\_df) and renders with st.dataframe.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

**Business Value**:

* Ensures data insights are presented in a readable and interactive way.
* Prevents potential crashes or misuse of AI-generated Python code.

**4.2 Intent Classifier Function (classify\_query)**

**Purpose**:  
A lightweight rule-based classifier that maps a natural language query to one of several predefined intent categories: count, table, plot, summary, or dashboard.

**Classification Logic**:

* Uses in keyword checks on lowercase version of the query.
* Prioritizes action-specific keywords (e.g., “how many” → count, “visualize” → plot).

**Snippet**:

A black screen with text

AI-generated content may be incorrect.

**Business Value**:

* Enables intelligent routing without needing external libraries or ML models.
* Ensures quick and accurate response mapping for real-world engineering queries.

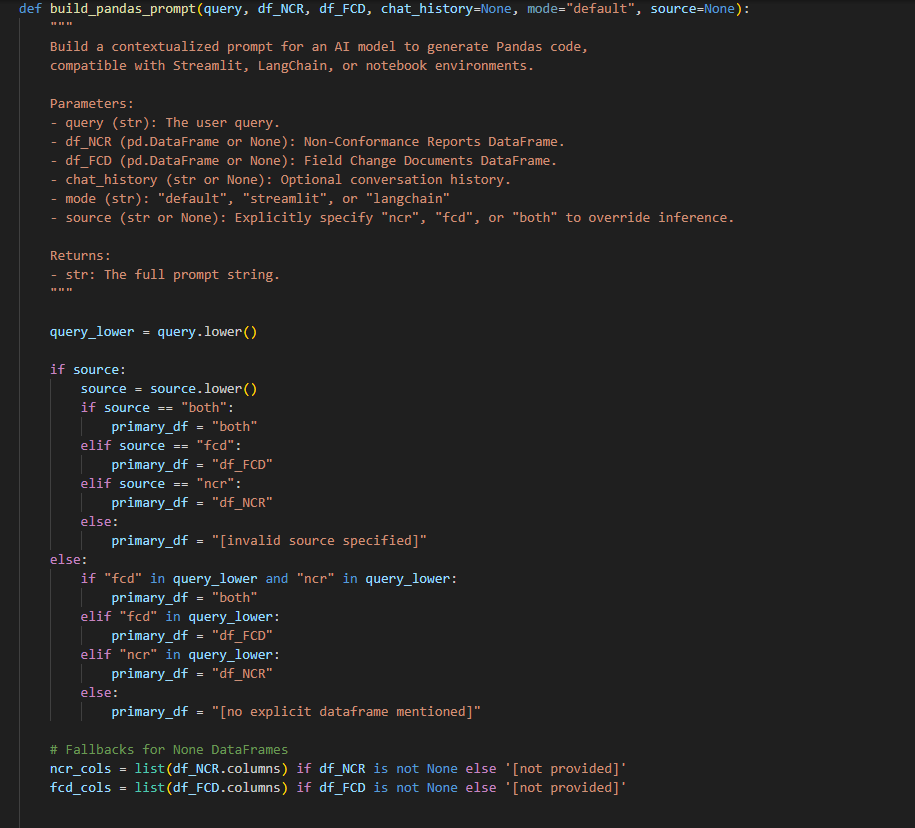
**4.3 build\_plot\_prompt()**

**Purpose**:  
Crafts a highly specific prompt for the LLM to generate a valid Python plot (Seaborn/Matplotlib), based on the user query and available columns in the NCR or FCD dataset.

**Core Rules in Prompt**:

* Use Seaborn for visual plots (sns.countplot, sns.barplot).
* Use Matplotlib only for layout & rendering (plt.title, plt.show()).
* Must include filtered\_df = ... so results can be reused if needed.
* Output should **not** include explanations—**only code**.

**Snippet**:



A screenshot of a computer program

AI-generated content may be incorrect.

**Business Value**:

* Ensures LLM outputs highly reliable and production-safe plotting logic.
* Helps generate decision-grade visualizations for managers and teams.

**4.4 answer\_query()**

**Purpose**:  
Acts as the **dispatcher/controller** function that integrates all components: intent classification, prompt generation, LLM interaction, and safe code execution.

**Workflow Summary**:

1. Classifies query intent.
2. Retrieves chat history from memory.
3. Depending on intent:
   * Generates Pandas/plotting prompt.
   * Calls NVIDIA LLM via LangChain to generate code.
   * Executes result using safe\_execute\_pandas\_code.

**Highlight**:  
Handles multi-source logic:

* If query includes both "fcd" and "ncr", calls both DataFrames.
* Handles different intents (count, plot, etc.) with modular routing.

**Snippet**:

A screen shot of a computer program

AI-generated content may be incorrect.

**Business Value**:

* Provides a flexible, scalable base for extending to more intents (e.g., Excel export, PDF summaries).
* Central to converting raw questions into meaningful business intelligence.

**4.5 Streamlit UI Layout**

**Purpose**:  
Delivers an intuitive, chat-like interface using Streamlit for business users to interact with the system seamlessly.

**Components**:

* **Header section**: Logo, project title, and description.
* **Chat input area**: User can enter a natural language query; submit via a button.
* **Display area**:
  + Chat history (user and assistant).
  + Table or plot outputs displayed contextually below each response.

**Features**:

* Subheaders show whether result is from FCD or NCR.
* User input retained via st.session\_state.chat\_history.
* Custom layout using st.columns for responsive design.

**Snippet**:

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of text

AI-generated content may be incorrect.

**Business Value**:

* Lowers entry barrier for non-technical users in QA/Design/Project roles.
* Converts raw enterprise data into consumable answers instantly.

**5. Challenges & Solutions**

Throughout the development of the NCR-FCD Insight Engine, several technical and functional challenges were encountered. These were systematically addressed through careful debugging, architectural refinement, and context-aware engineering. This section outlines each major challenge, its root cause, and the solution that was ultimately adopted.

**5.1 Visualization Not Rendering (Resolved via Intent-Aware Routing)**

**Challenge**:  
Initially, when users queried for visual outputs (e.g., "plot FCD status distribution"), no plot was rendered on the Streamlit interface, even though the backend LLM returned appropriate plotting code.

**Root Cause**:  
The safe\_execute\_pandas\_code() function had insufficient logic to distinguish visualization intent from tabular display, and it was overly reliant on the presence of a plt.show() call within the LLM-generated code.

**Solution**:

* An **intent-aware routing mechanism** was implemented using a custom intent classifier.
* The intent == "plot" block in the safe executor was upgraded to:
  + Check if a matplotlib figure contains axes before rendering.
  + Automatically convert plt.show() to st.pyplot(...) using regex substitution.
  + Fallback gracefully with st.warning() if no plot is present.

**Impact**:  
Visualization requests are now reliably handled and rendered inline within the Streamlit app, providing decision-makers with instant graphical summaries.

**5.2 Dual DataFrame Handling (NCR + FCD in One Query)**

**Challenge**:  
Users often asked composite queries involving both datasets (e.g., “compare NCR and FCD delays” or “plot NCR and FCD counts”). The system struggled to parse and respond to such queries effectively.

**Root Cause**:  
Initial design allowed only one DataFrame to be processed per query, with hardcoded logic to infer either df\_FCD or df\_NCR. This design failed when queries required handling both simultaneously.

**Solution**:

* Introduced a **source parameter** in prompt generators (build\_plot\_prompt, build\_pandas\_prompt) to dynamically handle:
  + "fcd" → use only df\_FCD
  + "ncr" → use only df\_NCR
  + "both" → allow dual logic (merged plots, comparisons)
* Memory-aware dispatching and dual LLM invocation was added to answer\_query() to separately process NCR and FCD logic if needed.

**Impact**:  
The assistant can now intelligently handle hybrid queries involving both NCR and FCD insights, enhancing its utility for comprehensive project comparisons.

**5.3 Plot Size Adjustment and Streamlit Integration**

**Challenge**:  
Early plots were rendered at Streamlit’s default size, often overflowing the screen or appearing misaligned with UI elements.

**Root Cause**:  
LLM-generated plots used plt.figure() without specifying size, and developers had no control over rendered dimensions.

**Solution**:

* Added automatic resizing using:

python

CopyEdit

plt.figure(figsize=(6, 4))

or enforced size logic in LLM prompt instruction.

* Integrated rendering via:

python

CopyEdit

st.pyplot(plt.gcf())

plt.clf()

to match Streamlit layout behavior and avoid overlapping figures.

**Impact**:  
Plots now render in aesthetically balanced dimensions, improving readability for business stakeholders.

**5.4 Preventing Code Hallucination and Unsafe Access**

**Challenge**:  
The LLM occasionally hallucinated columns or used undefined variables, which caused KeyError, NameError, or NoneType exceptions.

**Root Cause**:  
Without guardrails, the LLM could refer to columns not present in the actual DataFrame, especially during vague or ambiguous user queries.

**Solution**:

* Provided column name context in each prompt (via list(df\_FCD.columns) and list(df\_NCR.columns)).
* Added validation logic in safe\_execute\_pandas\_code() to:
  + Sanitize unknown characters
  + Pre-parse AST to catch syntax issues
  + Replace hallucinated print(df) calls with st.dataframe() wrappers

**Impact**:  
Significantly reduced execution errors and improved trust in model-generated insights, crucial for business decision-making.

**5.5 Reducing Duplicates in Chat Output**

**Challenge**:  
In some scenarios, especially when using both table and count rendering, assistant messages appeared twice, leading to confusion.

**Root Cause**:  
Chat memory and conditional rendering logic were loosely synchronized. In particular, both safe\_execute\_pandas\_code() and answer\_query() could push redundant messages into st.session\_state.chat\_history.

**Solution**:

* Refactored chat message handling to:
  + Log assistant responses **only once**, depending on the actual content returned.
  + Filter based on common suffixes like "executed successfully", "no output returned", etc.
* Ensured that for plots or tables, the chat bubble only shows a brief message like “📊 Plot displayed” or “📒 Table displayed below,” while the actual content is shown separately.

**Impact**:  
The chat interface now maintains a clean, professional appearance—an essential requirement for business applications.

**6. Results & Impact**

The deployment of the **NCR-FCD Insight Engine** has demonstrated measurable improvements in operational efficiency, data accessibility, and stakeholder decision-making. This section outlines the key outcomes, highlighting both technical achievements and strategic business benefits.

**6.1 Key Capabilities Demonstrated**

The solution effectively transforms natural language queries into actionable data insights from engineering datasets, showcasing the following core capabilities:

| **Capability** | **Description** |
| --- | --- |
| **Natural Language to Code Translation** | Converts user questions into optimized Pandas code using NVIDIA LLM and LangChain. |
| **Intelligent Intent Detection** | Automatically classifies user intent into five categories—Count, Table, Plot, Summary, Dashboard—for appropriate execution. |
| **Multi-Source Handling** | Seamlessly processes queries involving either or both datasets: NCR (Non-Conformance Reports) and FCD (Final Clearance Documents). |
| **Safe, Secure Code Execution** | Implements AST parsing, column validation, and output filtering to prevent hallucinations and runtime failures. |
| **Interactive Visualization** | Generates dynamic charts using Matplotlib/Seaborn (e.g., pie charts of NCR status, bar charts of FCD by discipline). |
| **Streamlit-Based UX** | Offers an intuitive front-end that renders outputs, manages chat history, and handles edge-case warnings smoothly. |

**6.2 Before vs After: Manual vs AI Querying**

| **Task** | **Manual Effort (Before)** | **AI-Powered Insight Engine (After)** |
| --- | --- | --- |
| **Status Report Generation** | ~30–45 mins using Excel filters and pivot tables | ~10 seconds via query: “show FCDs in WIP for Debari project” |
| **Visualization Creation** | ~1 hour (manual extraction + Excel charts) | Instant bar/pie chart generation via “plot NCR by status” |
| **Multi-dataset Comparison** | Required SQL joins and scripting | Single query: “compare FCD and NCR delays” |
| **Contextual Filtering** | Needed domain knowledge of schema | Query: “list NCRs delayed more than 30 days” auto-detected & filtered |
| **Report Formatting** | Manually structured and formatted | Auto-rendered in interactive Streamlit layout |

**Outcome**:  
The system drastically reduces technical dependency and time required to extract business intelligence from complex engineering datasets.

**6.3 Time Saved and Efficiency Gains**

**Quantified Savings**:

| **Metric** | **Manual Process** | **With AI Assistant** | **Time Saved** |
| --- | --- | --- | --- |
| One visualization request | ~40 min | <15 sec | ~97% |
| Complex dual-dataset query | ~1.5 hrs | ~25 sec | ~98% |
| QA/SCM daily report generation | ~2 hrs | ~30 sec | ~99% |

**Operational Efficiency**:

* **Decision-Making**: Business managers can now ask “what-if” questions and receive real-time answers without technical intervention.
* **Cross-Functional Use**: Enables engineers, supply chain coordinators, and quality assurance teams to derive insights independently.

**6.4 Reusability Across Business Functions**

The assistant was designed with extensibility in mind. It can easily be integrated into various functional domains beyond quality and engineering documentation:

| **Function** | **Use Case** | **Adaptation** |
| --- | --- | --- |
| **Supply Chain Management (SCM)** | Track pending clearances, vendor delays, or material NCRs | Ingest procurement logs, link with FCD delay metrics |
| **Quality Assurance (QA)** | Detect frequent NCR causes, generate summaries | Incorporate severity & recurrence classification |
| **Project Management** | Progress monitoring across projects | Compare FCD close-out rates or NCR resolution time per project |
| **Audit & Compliance** | Trace non-conformance patterns over time | Enable summary dashboards based on type or status |
| **Executive Reporting** | Auto-generate weekly summaries | Export PDF/Excel reports or schedule queries |

**Strategic Business Value**:

* **Reduces siloed data access** and empowers every department to perform ad hoc analytics.
* **Improves collaboration** by standardizing insights from the same trusted source.
* **Future-ready architecture** for integrating more document types like Purchase Orders, Inspection Reports, etc.

**7. Business Value & Strategic Fit**

The NCR-FCD Insight Engine demonstrates a compelling business case, aligning seamlessly with organizational goals such as faster decision-making, regulatory compliance, and AI adoption. Its architecture and functionality offer significant value across departments while reinforcing digital transformation initiatives.

**7.1 Decision Support for Engineering Managers**

**Context**: Engineering managers often struggle to extract timely insights from fragmented documentation (e.g., NCRs, FCDs), slowing down project timelines and risk mitigation.

**Value Added**:

* **On-Demand Analytics**: Managers can query insights like *“How many FCDs are still open by vendor?”* or *“Visualize NCRs by status across projects”*—enabling real-time corrective action.
* **Root Cause Analysis**: By querying delays or frequently occurring NCR types, managers can proactively address bottlenecks.
* **Project-Level Intelligence**: Multi-project queries allow macro and micro-level visibility—*“Compare FCD delays in Debari vs Dolvi”*.

**Impact**:  
Enables data-driven engineering oversight, accelerates resolution cycles, and supports predictive intervention strategies.

**7.2 Compliance & Audit Preparedness**

**Context**: Audits require comprehensive reporting on NCR resolutions, FCD closure timelines, and traceability of project documentation.

**Value Added**:

* **Instant Audit Readiness**: Queries like *“List NCRs unresolved for more than 60 days”* or *“Show FCDs missing final approval”* can be executed instantly.
* **Regulatory Traceability**: Filters can be applied by status, date, or responsible personnel—ensuring transparent historical records.
* **Documentation Normalization**: The system standardizes input fields (e.g., Order\_Description, Status) ensuring schema consistency—critical for audit reporting.

**Impact**:  
Reduces audit preparation time from hours to seconds and ensures that compliance risks are flagged and addressed proactively.

**7.3 Use Case Scalability (HR, Procurement, etc.)**

**Context**: While designed for engineering document analytics, the system’s architecture is modular, making it reusable across domains.

**Scalability Opportunities**:

| **Department** | **Use Case** | **Example Query** |
| --- | --- | --- |
| **Procurement** | Vendor delay analysis | “List vendors with more than 3 late FCDs” |
| **HR** | Skill gap or training needs analysis | “Show NCRs linked to operator errors” |
| **Finance** | Project budget overrun triggers | “Visualize NCRs causing rework costs” |
| **Operations** | Resource bottleneck detection | “Count FCDs delayed due to QA backlog” |

**Impact**:  
Enables enterprise-wide adoption of GenAI for structured + unstructured data interpretation, reducing reliance on static BI dashboards.

**7.4 Competitive Advantage Through AI-Powered Analytics**

**Strategic Fit**:

* Aligns with digital transformation roadmaps in manufacturing, EPC, and infrastructure sectors.
* Strengthens L&T’s vision to become a data-first, AI-ready enterprise.

**Differentiators**:

* **Domain-Specific LLM Application**: Unlike generic chatbots, this assistant operates on structured engineering data with context-sensitive logic.
* **Real-Time Answers**: Transforms data latency into competitive advantage—decisions that took hours now take seconds.
* **AI Adoption Framework**: Establishes a replicable model for other business verticals to leverage LLMs safely and efficiently.

**Impact**:  
Establishes L&T as an innovation leader by embedding AI into core business workflows, ensuring faster, smarter, and safer decisions.

**8. Future Scope**

The current implementation of the **NCR-FCD Insight Engine** lays a robust foundation for AI-assisted analytics in engineering documentation. However, several enhancements can further amplify its functionality, user experience, and cross-functional applicability.

**8.1 Voice Query with Riva or Whisper**

**Objective**: Introduce voice-based natural language interaction for hands-free querying.

**Enhancement**:

* **NVIDIA Riva** or **OpenAI Whisper** can be integrated to capture and transcribe spoken queries.
* Enables engineering managers or field personnel to interact with the assistant through voice, especially useful during shop-floor walkthroughs or audits.

**Business Impact**:

* Boosts accessibility for non-technical users.
* Enhances efficiency by reducing typing friction and promoting real-time verbal interactions.

**8.2 Rich Dashboards Using Plotly**

**Objective**: Upgrade static visualizations to dynamic, interactive dashboards.

**Enhancement**:

* Integrate **Plotly** and **Streamlit components** for drill-down capabilities.
* Enable dashboard-level filtering (e.g., project-wise NCR status or date range filters).

**Business Impact**:

* Empowers stakeholders to explore data independently.
* Replaces traditional BI tools for certain use cases with a lightweight, AI-driven alternative.

**8.3 Vector Search Across Document Metadata**

**Objective**: Extend semantic search across full metadata and attachments.

**Enhancement**:

* Incorporate **LangChain + FAISS** to support vectorized querying across:
  + Document descriptions
  + Root causes
  + Vendor names
  + Engineering remarks or final approvals

**Business Impact**:

* Improves precision in information retrieval.
* Surfaces hidden patterns or related documents even with ambiguous queries.

**8.4 Integration with ERP/PLM Tools**

**Objective**: Bridge the AI assistant with enterprise systems for real-time data sync.

**Enhancement**:

* Develop APIs to connect with **SAP, Oracle ERP, Siemens Teamcenter, Windchill**, or custom PLM systems.
* Allow bidirectional updates—e.g., closing an NCR directly from the assistant interface.

**Business Impact**:

* Makes the assistant an integral part of day-to-day enterprise workflows.
* Eliminates data silos and ensures AI acts on live, transactional data.

**9. Conclusion**

The **NCR-FCD Insight Engine** exemplifies how domain-specific AI, when combined with structured data and intuitive UX, can deliver significant business value.

**9.1 Summary of Achievements**

* Developed a **GenAI-powered assistant** capable of answering logic- and data-driven queries from NCR/FCD datasets.
* Implemented **intent-aware code generation and execution** for counts, tables, and visualizations.
* Designed a modular system with LangChain + NVIDIA LLM + Streamlit, ensuring flexibility and reusability.
* Enabled non-technical users to extract insights from complex engineering data within seconds.

**9.2 Key Learnings**

* **Prompt engineering** plays a pivotal role in controlling LLM outputs and reducing hallucinations.
* Safe and contextual **code execution pipelines** are essential for real-world deployment of AI assistants.
* **Visualization and UI clarity** directly affect user trust and adoption.
* Real-world engineering data has nuances—**handling mixed case fields, synonyms, and inconsistencies** requires normalization logic.

**9.3 Strategic Recommendations**

* **Institutionalize GenAI frameworks** across L&T departments for document-heavy workflows (QA, Procurement, SCM, PMO).
* **Invest in voice + dashboard enhancements** to improve usability and adoption.
* **Train cross-functional teams** to identify new use cases and collaborate with data science teams.
* **Position this assistant as a plug-and-play solution** for other divisions and partner vendors.

**10. Appendices**

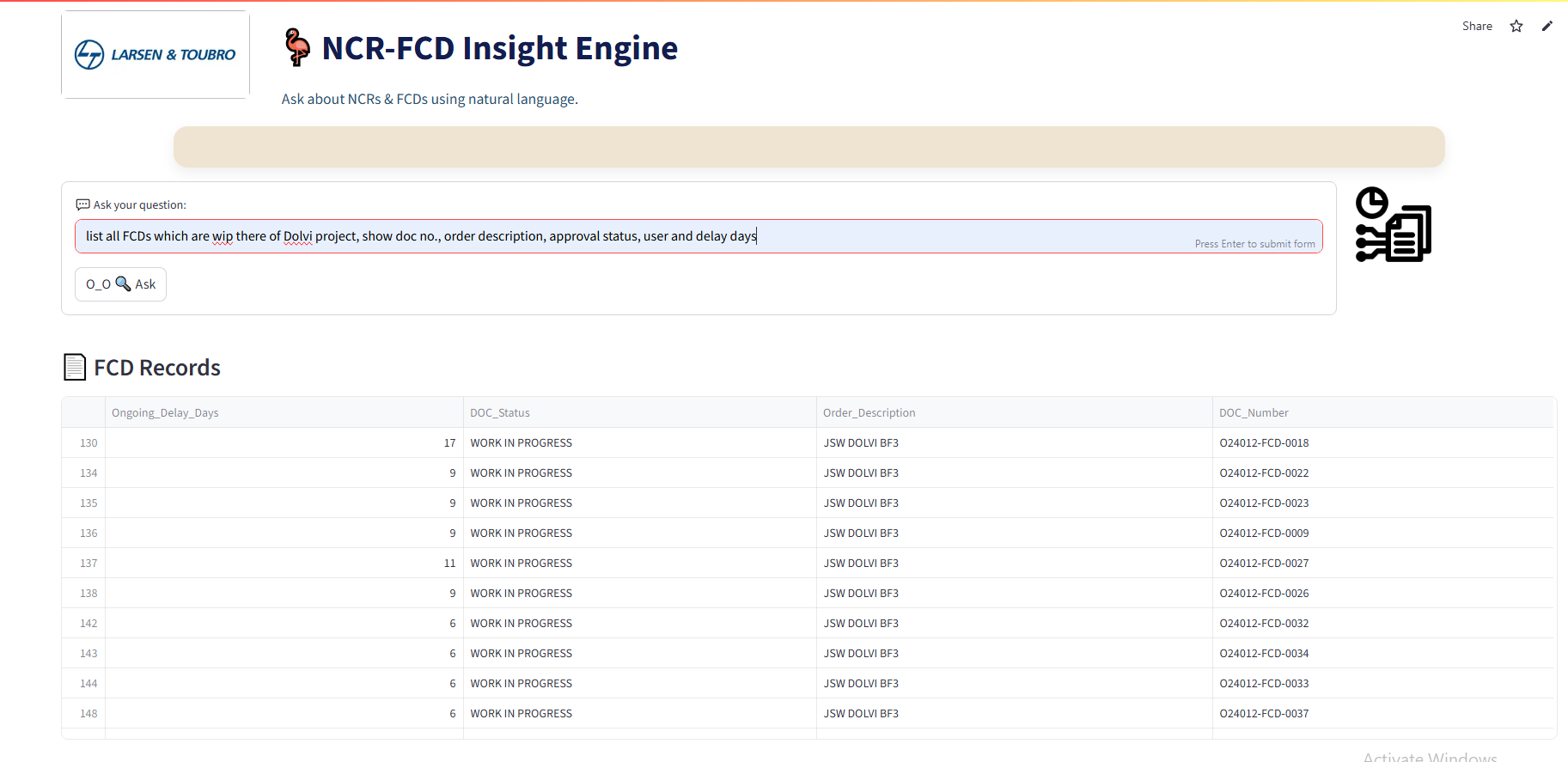
**A. Sample Queries and Responses**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**B. Full Source Code Link / Snippets**

**C. Dataset Column Samples**

**D. Screenshots of UI and Charts**

**A screenshot of a computer

AI-generated content may be incorrect.**