Technical Report: Call Compliance and Quality Analysis System

1. Introduction--

This project is about analyzing customer–agent conversations to check both compliance and quality. The idea is to see how well agents follow the rules and also how professional the conversations sound. The system focuses on three main areas:

- **Question 1:** Detecting profanity used by either the agent or the customer.
- **Question 2:** Checking if an agent ever shared sensitive account details *before* verifying the customer's identity.
- **Question 3:** Looking at call quality metrics by calculating silence and overtalk percentages.

The transcripts are in JSON format, with each line capturing who spoke (speaker), what they said (text), and when (stime, etime). Each file name acts as the call ID.

For Q1 and Q2(profanity detection and sensitive information leak detection), I tried two different approaches. First was a simple **regex-based method** that matches patterns. Then I explored **LLMs**, testing Qwen 1.5B and Meta LLaMA 1B Instruct. Both worked to some degree, but they weren't strong enough on compliance rules. Eventually, I settled on **Gemini** from Google, which consistently gave better and more structured results.

For Q3 (call quality metrics visualization), I built a second page in the Streamlit app that can either accept a ZIP of many calls or a pasted JSON prompt for a single one. On top of that, I also built a notebook to study aggregate insights like distributions and patterns across all calls of the given dataset.

2. System Design--

Regex Approach

The regex method is straightforward.

- I normalized all text using NFKC and unidecode so that tricks like writing 'c@#p' instead of 'crap' would still be caught (Since the given dataset is a recorded call converted to text, chances of such scenarios are less. But still, this is a safe plan implemented to make sure all profanities in text are detected).
- I wrote regex rules for curse words and for Personal Identifiable Information (PII) (such as DOB, SSN, account numbers, etc.).
- The logic flags when profanity appears and whether sensitive details were shared *before* verification.

This method is cheap, fast, and very reliable for fixed patterns, but it breaks down when the language changes or when context is needed.

LLM Approach

I initially tried Qwen 1.5B Instruct and Meta LLaMA 1B Instruct locally. They were fine for simple text classification but not strong enough for compliance checks, especially where context mattered (like figuring out whether disclosure came before or after verification).

Switching to Gemini improved the results significantly. I could define the compliance rules in the system prompt, and the outputs came back in a structured way that was easier to parse into CSV.

So, in short: regex gives you speed and determinism, while Gemini gives you flexibility and context-awareness, even if it's slower and dependent on an external API.

Visualization and Metrics (Q3)

For Q3 (visualization of call quality metrics), the system looks at utterance timestamps:

- It calculates overtalk whenever agent and customer speak at the same time.
- It calculates silence whenever nobody is speaking.
- There's a second Streamlit page where you can upload data and see a bar chart for each call.
- There's also a notebook that aggregates across many calls to see overall trends.

3. Implementation Recommendations--

Q1: Profanity Detection

Regex:

- Perfect for standard swear words and common obfuscations.
- Runs quickly, works in real time, and scales easily.
- The drawback is it can't keep up with slang or indirect offensive language.

LLM (Gemini):

- Picks up on indirect profanity, sarcasm, or coded language.
- More flexible as language evolves.
- Slower, more expensive, and sometimes inconsistent.

My take: For large-scale production, regex should be the main solution. It's fast and reliable. LLMs are best used as an audit layer or when you specifically want to analyze tone or subtlety.

Q2: Privacy and Compliance Violation

Regex:

- Very good at spotting raw PII like DOB or account numbers.
- But regex alone cannot tell whether the agent verified the customer before disclosing information.

LLM (Gemini):

- Strong at understanding sequence and context.
- Can correctly flag a violation if disclosure came before verification.
- Needs careful prompt design and validation to minimize mistakes.

My take: For compliance (where context is critical), the LLM should be the primary tool. Regex can still be useful as a quick pre-filter to highlight transcripts that contain PII.

.

4. Visualization Analysis (Q3)--

I built both a Streamlit page and a notebook for call quality metrics. The Streamlit page helps with per-call inspection, while the notebook provides organization-level insights that can help in understanding the agent-customer relation progress and quality.

Key Insights(Specifically for the given conversation):

1. **Weighted vs. Average Percentages**
    - Overtalk is consistently around 10% across calls.
    - Silence is lower, at about 3–4%.
    - Overtalk is clearly the more frequent issue.
2. **Distribution of Overtalk**
    - Most calls have 8–15% overtalk.
    - Mean and median are close, showing this is normal rather than driven by outliers.
    - A few extreme calls exceed 30–40% overtalk, which deserve closer review.
3. **Distribution of Silence**
    - Silence is very low in most calls (median 1.3%).
    - Outliers with 8–12% silence may indicate disengagement or long holds.
4. **Call Duration vs. Overtalk**
    - Short calls (<40s) sometimes spike with high overtalk.
    - Longer calls (60–100s) settle around 5–15% overtalk.
    - Suggests rushed, overlapping starts that smooth out as the call goes on.

Recommendation: These metrics should be included in QA dashboards. Outlier calls with extreme silence or overtalk should be flagged for coaching. Over time, tracking these numbers can help identify systemic training needs.

5. Running the LLM Notebook

To run the Gemini notebook (Profane&Compliance(LLM)) locally:

1. Go to Google AI Studio and generate an API key.
2. Copy that key into your notebook:

   genai.configure(api_key="...") → Replace "..." with your API key

3. Install the dependencies:

   pip install -r requirements.txt

4. Open and run "Profane&Compliance(LLM).ipynb". It will:
   - Load call transcripts.
   - Send them to Gemini using the system prompt.
   - Parse and save outputs into outputs/gemini_summary.csv**.**

Note: Running the other two notebooks (Profanity&Compliance(RegEx).ipynb and the Call_Quality_Metrics_Analysis.ipynb) only requires the dependencies listed in 'requirements.txt'.

6. Running the Streamlit App

The Streamlit web app has been deployed to the cloud and can be accessed directly through the link (https://profanity-compliance-detector-0.streamlit.app/).

- The app requires two CSV files to work correctly:
  - outputs/gemini_summary.csv:

    generated from Profane&Compliance(LLM).ipynb
  - outputs/regex_summary.csv:

    generated from Profanity&Compliance(RegEx).ipynb
- Once these CSVs are generated in the environment, the main page allows you to view results for Q1 (profanity) and Q2 (compliance violations).
- The second page (Q3 Visualizer) lets you upload a ZIP of calls or paste a JSON prompt to view silence and overtalk metrics.

7. Conclusion--

This project shows how regex and LLMs can work together to analyze call-centre transcripts. Regex is fast, cheap, and perfect for fixed rules like profanity detection or spotting raw PII. LLMs add the ability to reason about context, which is essential for compliance checks.

In practice, regex should handle high-confidence cases, while LLMs are best for edge cases where order and meaning matter. Together, they form a hybrid pipeline that balances efficiency and accuracy.

My own experiments confirmed this. When I tested Qwen 1.5B and Meta LLaMA 1B Instruct, they struggled with compliance rules. Gemini performed far better, giving structured, consistent outputs. This shows that model choice really matters.

Exploring other LLMs, whether smaller domain-specific models or newer state-of-the-art ones, it could improve both accuracy and reliability. Fine-tuning on domain-specific data or building ensemble methods (regex + multiple models cross-checking) could push the system further.

In summary, this project laid out a practical foundation for real-world compliance monitoring: regex for speed and precision, LLMs for context, and visualization for quality insights.