LogLens

A Virtual Assistant for Log Analysis and System Health

Leveraging large language models to efficiently query SQL databases and the web.

This NLP class project aims to simplify system log access and monitoring.

A **by Aditya Rao**

Project Motivation

- **24/7 Service Ownership**

  Engineers must constantly track latency spikes, throughput drops, and error surges across distributed services.

- **Log Overload**

  Millions of log lines per hour make manual searches and ad-hoc SQL queries impractical and error-prone.

- **Faster Troubleshooting**

  Every minute spent hunting log patterns delays incident resolution—hurting SLAs and user satisfaction.

- **Smart Automation**

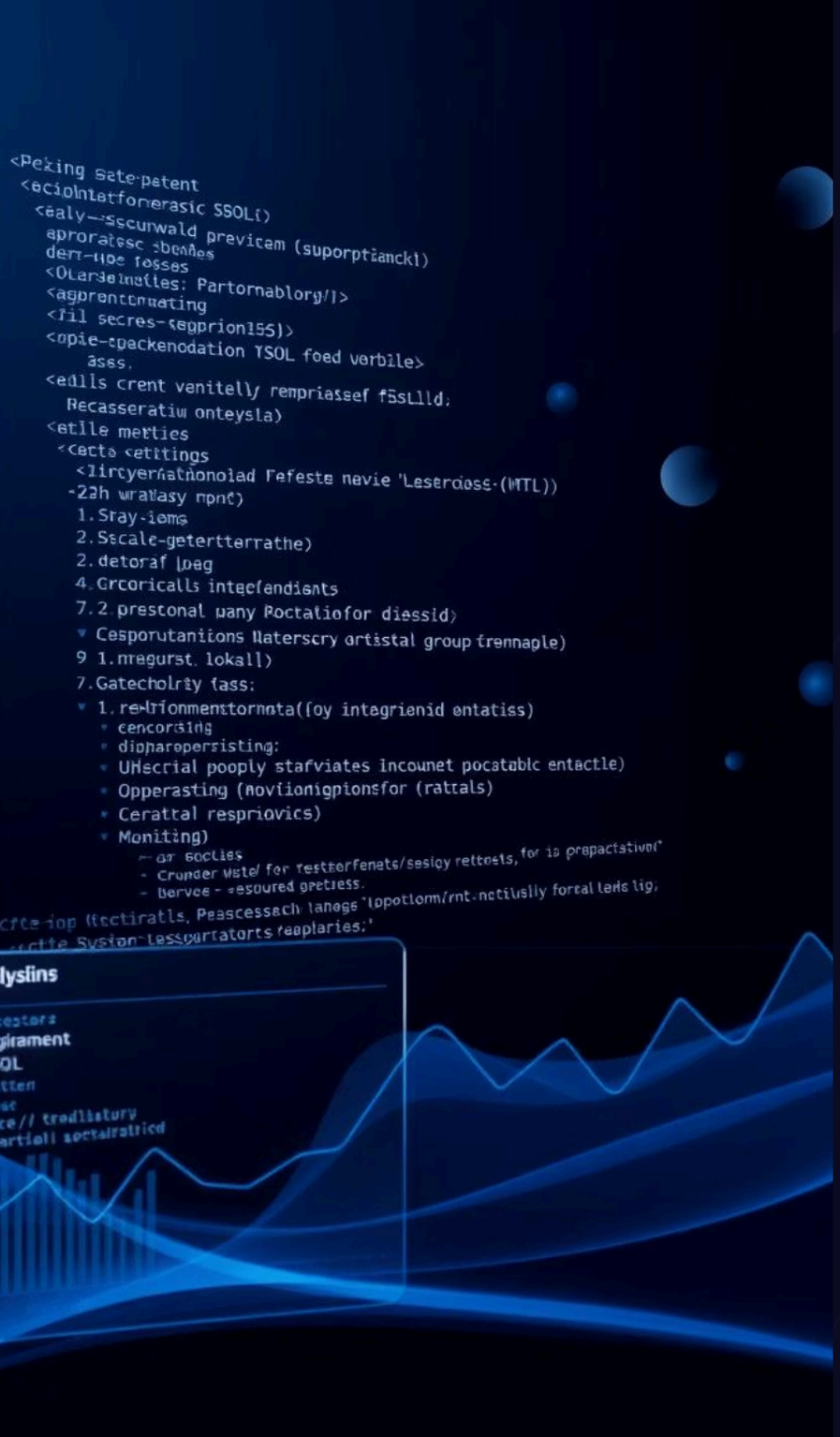
  A Virtual Assistant that:

  a. **Parses** terabytes of logs and generates precise SQL on demand

  b. **Infers** actionable SRE advice (e.g. tune connection pooling)

  c. **Surfaces** curated web-search results for best practices

Empower teams to spot anomalies instantly, reduce MTTR, and stay ahead of system health issues.



Project Overview and Objectives

SQL Query Generation

Convert natural language to SQL for detailed log analysis.

System Health Q&A

Answer user questions on CPU, memory, and disk metrics.

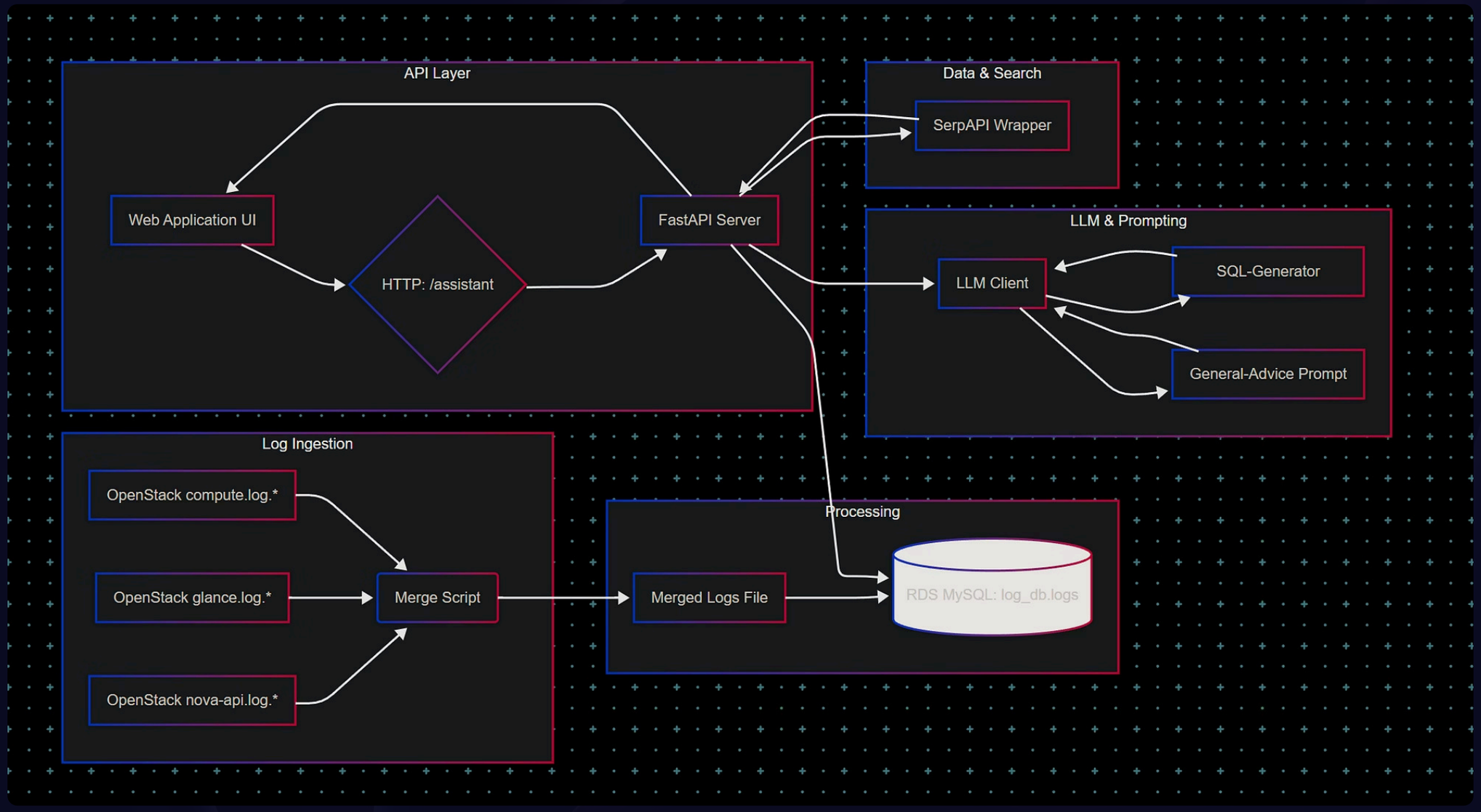
Web Search Integration

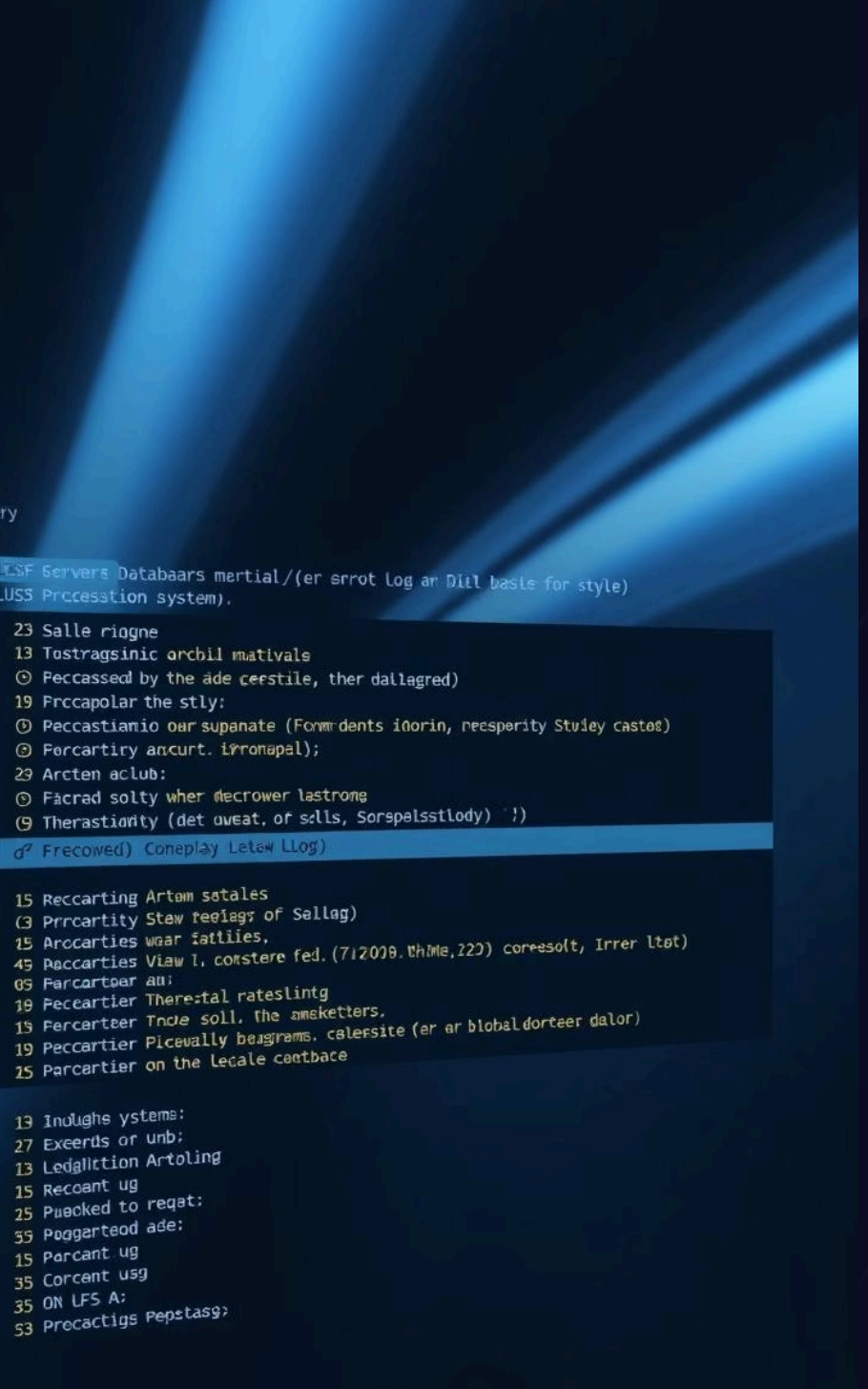
Fetch relevant external info using web search APIs like Google.

Technologies Used

LLMs, SQL database, web search API for robust interaction.

System Architecture





Functionality 1: Log Analysis via SQL Queries

1 User Query

Example: Find anomalous log messages within a date range.

2 LLM Role

Translate question into SQL syntax accurately.

3 SQL Example

SELECT * FROM logs WHERE log_level = 'ERROR' AND timestamp BETWEEN dates;

4 Output

Display retrieved log entries for user review.

Functionality 2: System Health Q&A

1

User Query

Example: How to improve system health metrics?

2

Interpretation

LLM understands question context and gives response as a site reliability expert.

3

Security

Prompt injection techniques are handled to prevent access to sensitive data.

4

Response

Text based response based on the user input.

Functionality 3: Web Search Integration

SerpAPI integration

User inputs query in natural language

Web Search API

Google Search API fetches relevant documents.

Result Presentation

Answers is presented with an overview and associated links.

Advanced prompting techniques

Schema Injection

Embed the full logs table schema in the system prompt so the model always knows column names and types.

SQL-Only Enforcement

Instruct the model to output *only* the SQL statement (no extra commentary)

Few-Shot Examples

Provide 2–3 input/output SQL examples in the prompt to teach the model your desired query patterns.

Dynamic Context Injection

Prepend the last 5 recent log entries as `### RECENT LOG EXAMPLES:` in the system prompt to ground generated queries in real data.

Chain-of-Thought Prompting

For SRE advice, use a “think step by step” system prompt so the model returns a structured, multi-point recommendation.

Prompt Caching

Wrap `sql_from_nl` and `ask_general` in an LRU cache (128 entries) to instantly replay repeat prompts without re-calling the LLM.

Prompt-Injection Detection

Scan incoming queries for risky patterns (e.g. “ignore all previous instructions”) and refuse malicious requests while preserving the uniform response schema.

Model Selection Logic

Dynamically choose between multiple providers/LLM endpoints based on the `model` name in the request.

Model Comparison: Cohere vs Qwen

Prompt: Return count of all logs where APIs were successful vs failures in 2017

Cohere Model - 111b parameters (CohereLabs/c4ai-command-a-03-2025)

Aggregates success and failure counts with concise SQL using conditional sums.

- Generated SQL : SELECT \n SUM(CASE WHEN log_status BETWEEN 200 AND 299 THEN 1 ELSE 0 END) AS successful_requests,\n SUM(CASE WHEN log_status BETWEEN 400 AND 599 THEN 1 ELSE 0 END) AS failed_requests\nFROM logs\nWHERE YEAR(log_timestamp) = 2017;
- More sophisticated SQL query
- Accurate result on exact comparison values

Qwen Model - 7b parameters (Qwen/Qwen2.5-VL-7B-Instruct)

Groups logs by each status code with detailed counts per code.

- Generated SQL : SELECT log_status, COUNT(*) FROM logs WHERE log_timestamp BETWEEN '2017-01-01' AND '2017-12-31' GROUP BY log_status;
- Less sophisticated SQL query
- Outputs multi-row result with counts per status - not accurate

Conclusion and Future Work

Summary

Assistant skillfully answers logs, system health, and searches.

Future Enhancements

- Understand more complex context between services and give response on it.
- Predict anomalous log patterns in advance.
- Add proactive alert notifications based on logs

Q&A

Open floor for questions and discussion.