Leveraging Generative AI for L2 Support in Python Applications

# 

# Objective

Implement L2 support using LLMs (LLaMA 3 / similar models) to automate root cause analysis from logs of Python applications, improve reliability, and reduce resolution time.

# Expected Capabilities

* Error Summarization: Automatically extract and summarize unique errors from logs.
* Root Cause Identification: Identify reasons behind errors using context-aware understanding.
* Solution Recommendation: Suggest probable solutions based on similar patterns and known fixes.
* Error Trend Analysis: Analyze logs over time to provide preventive insights and error rate reduction strategies.
* Cloud Integration: Seamlessly integrate with cloud platforms like GCP, Azure, or AWS for scalable deployment.
* Dashboard Visualization: Provide real-time insights via Grafana Dashboards (error frequency, resolution time, model recommendations, system health).

# Additional Business Benefits

* Reduced MTTR (Mean Time to Resolution): Quickly identify issues without manual log analysis.
* Knowledge Retention: LLM retains solutions and context from past issues—building an internal knowledge base.
* Lower Support Costs: Automates L2 tasks, freeing up engineering teams for high-value work.
* Scalability: Supports thousands of logs across multiple services in real-time.
* Decision-Making with Data: Visual dashboards aid in strategic decisions around code quality and team productivity.

# Grafana Dashboard: Sample Visualization Layout

|  |  |
| --- | --- |
| Panel | Description |
| Error Trends | Line chart showing error frequency over time |
| Top 10 Errors | Bar graph of most frequent errors |
| Root Causes | Table or pie chart summarizing categorized reasons |
| Solution Outcomes | Success rate of auto-suggested resolutions |
| Cloud Usage Metrics | Logs processed per cloud region/service |

# Next Steps

1. Choose LLM (LLaMA 3 / OpenAI / Mistral)
2. Define log input format
3. Build and test prompt templates
4. Integrate with monitoring tools & cloud
5. Create Grafana dashboards
6. Measure success and iterate