📌 Prompt Template for Generating Day 1 Observability Training (Markdown Format)

🎯 Purpose

This prompt is used to instruct a GPT model (4.0, 4.5, or fine-tuned variants) to generate **structured**, **diagram-enhanced**, **incident-aware** SRE training content on the Three Pillars of Observability: **Metrics**, **Logs**, and **Traces**.

Use this in your GPT UI as a full pasteable prompt. All variables marked like `{{SECTION}}` or `{{VIDEO_LINK}}` are to be filled manually or by scripting.

📝 Prompt Start

You are an experienced SRE instructor writing a Day 1 training module on observability for an audience of beginner to intermediate DevOps engineers. This training must:

💡 Required Characteristics

- Be structured by the **Three Pillars**: Metrics, Logs, Traces
- Include **Mermaid diagrams** per section to illustrate flows and architecture
- Provide a **realistic production incident story** per pillar (a.k.a. "horror story with happy ending")
- Include **Python code samples** for each pillar
- Offer **tiered learning objectives** for: 🔍 Beginner, 🧩 Intermediate, 💡 Advanced/SRE
- Clearly indicate **where curated YouTube videos** should be inserted
- Provide actionable, non-generic, step-by-step explanations
- Use consistent Markdown formatting, emoji tier tags, and section breaks
- Encourage humor or empathy when describing real-world issues

---

🧱 Prompt Structure

📌 1. Introduction

```
## 📌 Introduction: Observability 101

- Explain observability using the "Observe, Test, Evaluate, Take Action" framework
- Clarify how observability differs from monitoring
- Use a visual metaphor: observability as a diagnostic triage room
- Mermaid Diagram: Three Pillars Flow → Detection → Resolution
- 🔥 Incident Story: An alert was firing; logs were unclear; metrics saved the day
- 📺 YouTube: {{VIDEO_LINK_INTRO}}
```

📊 2. Metrics

```
## 📊 Metrics: The Quantified View

- 🔍 Beginner: Counters, Gauges, Histograms with Prometheus
- 🧩 Intermediate: RED method, custom metrics, visualization
- 💡 SRE: Alert tuning, cardinality concerns, data pipeline issues
- Mermaid Diagram: Metrics flow from Flask → Prometheus → Grafana
- 🔧 Code Example: Prometheus + Python Flask
- 🔥 Horror Story: Cardinality explosion brought Prometheus to its knees
- 📺 YouTube: {{VIDEO_LINK_METRICS}}
```

🪵 3. Logs

```
## 🪵 Logs: The Narrative Thread

- 🔍 Beginner: Log levels and basic logging
- 🧩 Intermediate: Structured JSON logs, log aggregation
- 💡 SRE: Querying logs for trace IDs, correlating error chains
- Mermaid Diagram: App → FluentBit → Elasticsearch → Kibana
- 🔧 Code Example: Python + structlog integration
- 🔥 Horror Story: Grepping for errors for 4 hours because no one added
request_id
- 📺 YouTube: {{VIDEO_LINK_LOGS}}
```

🧿 4. Traces

```
## 🧿 Traces: The Request's Journey

- 🔍 Beginner: Spans, trace IDs, visual timelines
- 🧩 Intermediate: Adding tracing to Flask with OpenTelemetry
- 💡 SRE: Context propagation across microservices
- Mermaid Diagram: Request spans from API → Service → DB
- 🔧 Code Example: Flask + OpenTelemetry + Jaeger
- 🔥 Horror Story: 5s checkout traced to a forgotten microservice timeout
- 📺 YouTube: {{VIDEO_LINK_TRACES}}
```

🔄 5. Pillar Integration

```
## 🔄 Integrating the Three Pillars

- Show how metrics, logs, and traces correlate via trace_id
- Mermaid Diagram: Full flow including Prometheus, Kibana, Jaeger
- 🔧 Code: Python Flask app emitting all three observability signals
- 🔍 Tip Box: Link dashboards using shared metadata (e.g. trace_id)
- 📺 YouTube: {{VIDEO_LINK_INTEGRATION}}
```

🎮 6. Hands-On Tiered Challenges

```
## 🎮 Hands-On Labs

- 🔍 Beginner: Instrument Flask with Prometheus metrics
- 🧩 Intermediate: Add JSON logs and build Kibana queries
- 💡 SRE: Correlate a trace from Jaeger with matching logs and metrics
- Bonus: Inject a bug and trace it across the stack
```

🔥 7. Real-World Incident Stories

```
## 🔥 Incident Walkthroughs

### Scenario 1: Performance Regression
- Metrics: Latency spike
- Logs: DB timeouts
- Trace: Span delay in service B → C
- Resolution: Reduced DB pool size
- 📺 YouTube: {{VIDEO_LINK_SCENARIO1}}

### Scenario 2: Intermittent 500s
- Metrics: Steady 0.1% error rate
- Logs: Null pointer exception with user-agent
- Trace: Correlated only with long-lived sessions
- Resolution: Added header validation
- 📺 YouTube: {{VIDEO_LINK_SCENARIO2}}
```

⚙️ Diagram Policy

- Diagrams must use proper Mermaid syntax
- Always include at least one diagram per major section
- Use quotes around node labels with `<br/>`, `:` or special characters

☑️ Markdown Format Rules

- Emojis for tier: 🔍, 🧩, 💡
- Code blocks use syntax highlighting (```` ```python ````)
- Tables for comparisons where possible
- Use real timestamps and structured labels in log examples

🧠 Invocation Summary

Generate a detailed, visually rich Markdown document teaching the Three Pillars of Observability. Each section must include:

- Code example (Python/Flask)

- Mermaid diagram
- Horror story with fix
- Learning objectives per tier
- YouTube placeholder

The goal: real training that's **ready for production onboarding** and not a lukewarm blog post.

Prompt ends here. Output must be Markdown only.