“Multimodal AI For Network Operations Centers”

A PROJECT REPORT

Submitted by

Gowripriya R (DL.AI.U4AID24113)
Yaalini R (DL.AI.U4AID24043)
Vepuri Satya Krishna (DL.AI.U4AID24140)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY (B. Tech) IN ARTIFICIAL INTELLIGENCE AND DATA SCIENCE (AIDS)

Under the guidance of

Dr. Divyanshu Sinha
Dr. Vivek Patel

Submitted to

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

FARIDABAD – 121002

December 2025

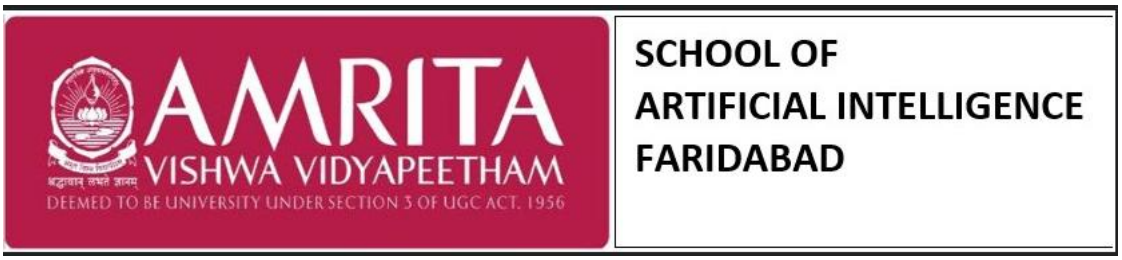
AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956

SCHOOL OF
ARTIFICIAL INTELLIGENCE
FARIDABAD

BONAFIDE CERTIFICATE

This is to certify this project report entitled "**Multimodal AI For Network Operations Centers**" is the Bonafide work of " **Gowripriya R (DL.AI.U4AID24113)**, **Yaalini R (DL.AI.U4AID24043)** and **Vepuri Satya Krishna (DL.AI.U4AID24140)**" , who carried out the project work under my supervision.

Signature
Dr.Divyanshu Sinha
Dr. Vivek Patel
Assistant Professors
School of AI
Faridabad



DECLARATION BY THE CANDIDATE

I declare that the report entitled "**Multimodal AI For Network Operations Centers**" submitted by me for the degree of Bachelor of Technology is the record of the project work carried out by me under the guidance of "**Dr. Divyanshu Sinha and Dr. Vivek Patel**" and this work has not formed the basis for the award of any degree, diploma, associateship, fellowship, titled in this or any other University or other similar institution of higher learning.

SIGNATURE

Gowripriya R (DL.AI.U4AID24113)
Yaalini R (DL.AI.U4AID24043)
Vepuri Satya Krishna(DL.AI.U4AID24040)

ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our honorable Chancellor **Sri Mata Amritanandamayi Devi**, for her blessings and for being a source of inspiration. I am indebted to extend my gratitude to our Principal, **Dr. Lakshmi Mohandas** Amrita School of Computing and Engineering, for facilitating us all the facilities and extended support to gain valuable education and learning experience.

I wish to express my sincere gratitude to my supervisors "**Dr. Divyanshu Sinha and Dr. Vivek Patel**" for his personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Supervisor, Review Panel Members, and the entire faculty of the Department of Computer Science & Engineering, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

Gowripriya R (DL.AI.U4AID24113)
Yaalini R (DL.AI.U4AID24043)
Vepuri Satya Krishna(DL.AI.U4AID24040)

TABLE OF CONTENTS

S.no	Contents	Page. no
1.	Introduction	**1**
2.	Objectives	**2**
3.	Methodology/ Experiment/ Work Done	**3**
4.	Results & Discussion	**5**
5.	Conclusion	**6**
6.	References	**7**
7.	Appendix	**7**

Introduction

In today's world, computer networks are an important part of everyday life. Activities such as using mobile apps, attending online classes, accessing cloud services, and running large organizations all depend on stable and continuous network connectivity. As networks become larger and more complex, managing and monitoring them has become difficult. This work is handled by a Network Operations Center (NOC), which acts as a central unit for monitoring and maintaining network systems.

A Network Operations Center works 24 hours a day and continuously monitors routers, switches, servers, cloud resources, and applications. The main responsibilities of a NOC include identifying network faults, monitoring performance, responding to incidents, and making sure service level agreements (SLAs) are maintained. Traditional NOCs use multiple monitoring tools that generate large amounts of data such as system logs, performance metrics, alerts, and network topology information. These data sources are usually analyzed separately, which makes troubleshooting slow and highly dependent on human effort.

One of the major challenges in traditional NOCs is alert overload. A single network problem can produce a large number of alerts from different tools, making it hard for engineers to find the actual cause of the issue. Manually analyzing logs, metrics, and alerts takes time and can delay problem resolution, leading to increased downtime and reduced service quality. As networks continue to grow, these challenges show the need for smarter and more automated solutions.

This project focuses on the use of Multimodal Artificial Intelligence in Network Operations Centers. Multimodal AI refers to systems that can process and combine different types of data such as text, numerical data, and event information. In a NOC environment, multimodal AI can analyze logs, performance metrics, alerts, and network topology together to better understand network behavior.

By using transformer based multimodal models, the system can detect unusual patterns, identify anomalies, and provide useful insights about network issues. This helps NOC engineers find root causes faster, reduce unnecessary alerts, and take preventive actions before major failures occur. The aim of this project is to study how multimodal AI can improve the efficiency, reliability, and overall performance of modern Network Operations Centers.

Objectives

The main objective of this project is to understand the role of Multimodal Artificial Intelligence in Network Operations Centers and how it helps improve network monitoring and overall efficiency. Network Operations Centers continuously monitor network infrastructure to ensure proper performance, availability, and security. Since modern networks generate large amounts of different types of data, analyzing this data effectively has become very important.

One objective of this project is to understand how a Network Operations Center works. This includes learning how NOCs function as centralized control units that operate throughout the day to monitor routers, switches, servers, cloud resources, and applications. The project focuses on basic NOC activities such as monitoring networks, detecting incidents, troubleshooting problems, analyzing performance, maintaining systems, and handling escalations.

Another objective is to identify the problems faced by traditional Network Operations Centers. These include alert overload, use of multiple disconnected monitoring tools, and manual analysis of logs and events. Such issues make it difficult to quickly find the root cause of network problems and often result in delays and increased network downtime.

The project also aims to understand the concept of Multimodal Artificial Intelligence and its use in NOC environments. Multimodal AI combines different types of data such as system logs, performance metrics, alerts, network topology information, and operator inputs. Studying how these data sources work together helps in gaining better understanding of network conditions.

Another objective of this project is to study how Multimodal AI supports NOC teams in their daily tasks. AI based systems help in detecting anomalies in real time, predicting failures, classifying incidents automatically, and prioritizing important issues. This helps NOC engineers make faster and better decisions.

Finally, this project aims to study the advantages of using Multimodal Artificial Intelligence in Network Operations Centers. These include faster identification of root causes, reduced network downtime, better compliance with service level agreements, improved efficiency of NOC teams, and increased reliability of network systems. This project highlights how Multimodal AI can be a useful solution for modern network operations

Methodology

The methodology of this project is based on studying how Multimodal Artificial Intelligence can be used in Network Operations Centers to improve network monitoring and fault detection. The approach follows the normal working process of a NOC and explains how different types of network data can be analyzed together using AI.

The first step of the methodology is to understand the different types of data available in a Network Operations Center. A NOC continuously produces large amounts of data from sources such as system and application logs, performance metrics like CPU and memory usage, network alerts and alarms, network topology information, and operator notes. These different data sources are important for understanding the overall condition of the network.

The next step involves collecting and preparing the data for analysis. Data from monitoring tools is organized before being given to the AI model. This includes arranging log messages, structuring performance metrics, removing unnecessary alerts, and formatting topology information. This step makes sure that all data is in a suitable form for further processing.

After preprocessing, the system applies a multimodal AI approach that combines all types of data into a single model. Instead of analyzing logs, metrics, and alerts separately, the multimodal model processes them together. This helps in understanding how different data sources are related and allows the system to detect patterns that may not be visible when data is analyzed individually.

Transformer based models are used in this methodology because they are effective in handling different types of data together. Transformers use attention mechanisms to learn important relationships between logs, metrics, alerts, and topology information. This helps the model focus on important signals while ignoring less useful information. These models are also suitable for handling large scale network data.

Using multimodal fusion allows the system to analyze network behavior more accurately. For example, a network issue can be better understood when high latency values are analyzed along with error messages and topology details. This combined analysis helps in identifying faults more clearly and reduces confusion during diagnosis.

Once the data is combined, the system performs anomaly detection. The multimodal AI model identifies unusual patterns in the network and classifies events based on their severity. This helps in faster identification of problems and provides useful information to NOC teams.

Finally, the results of the AI system are presented to NOC operators through alerts and dashboards. This helps engineers focus on important issues, reduce manual analysis, and take timely actions. Overall, this methodology shows how multimodal AI can improve the efficiency and reliability of Network Operations Centers.

Prototype Implementation Using Simulated Data

Since real Network Operations Center data is confidential and not easily available, simulated network data was used in this project to demonstrate the working of the system. The simulated data is designed to resemble the type of information normally handled in a NOC.

The prototype uses different types of data such as system log messages, network performance metrics, alert levels, and a simple network topology. Log messages include common network events such as normal operations, warnings, errors, and critical failures. Performance metrics like latency, bandwidth usage, and packet loss are generated to represent both normal and faulty network conditions. Alert levels indicate the seriousness of each network event.

The log messages are processed by converting words into numerical values so that they can be understood by the model. Performance metrics are also converted into numerical vectors using basic mathematical transformations. Alert values are encoded in a similar way. This ensures that all data types are represented in a common format.

After preprocessing, all the data is combined and passed into a multimodal transformer model. The transformer uses attention to learn relationships between logs, metrics, and alerts at the same time. The final output of the model classifies each network event as either normal or anomalous. This prototype helps in understanding how multimodal AI can analyze different network data together instead of processing each type separately.

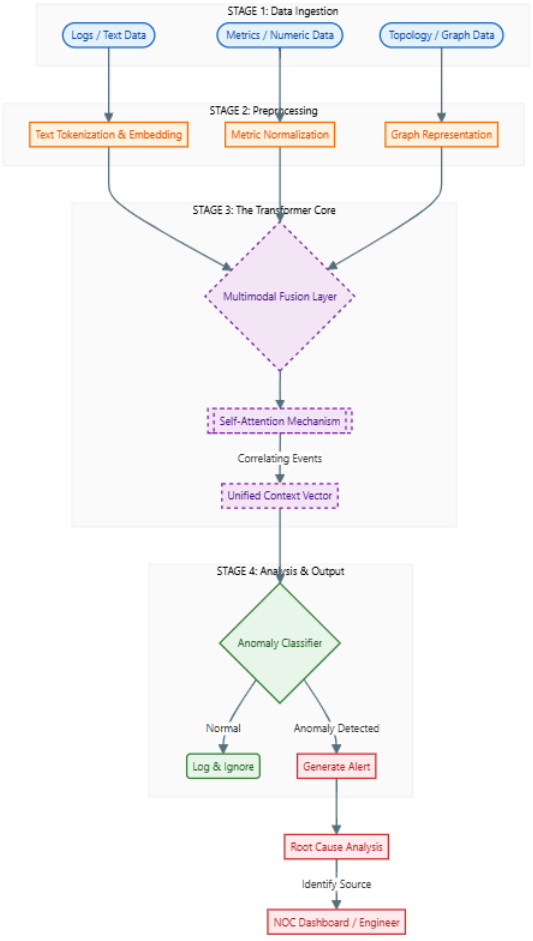


Figure 1: Architecture flow from data ingestion to actionable alerts.

Results and Discussion

This project studies how Multimodal Artificial Intelligence can be used in Network Operations Centers and how it affects network monitoring and fault handling. Based on the concepts discussed in the project, the results show that using multimodal AI improves the overall working of NOC operations when compared to traditional monitoring methods.

One of the main observations from this study is the improvement in identifying the root cause of network problems. In traditional NOCs, engineers manually analyze logs, alerts, and performance metrics, which takes time and can delay problem resolution. In the multimodal approach, logs, metrics, alerts, and topology information are analyzed together. This helps in finding the actual cause of network issues faster and reduces network downtime.

Another important result is the reduction in alert overload. In traditional systems, a single network problem can generate many alerts, which makes it difficult for operators to understand which alert is important. Multimodal AI helps by combining information from different data sources and highlighting only relevant alerts. This reduces alert fatigue and helps engineers focus on critical issues.

The study also shows that multimodal AI improves anomaly detection in network operations. By analyzing performance metrics along with log messages and alerts, the system can detect unusual behavior more effectively. This helps in identifying potential problems early and allows NOC teams to take preventive action before major failures occur.

Multimodal AI also supports better decision making for NOC teams. Since the system provides related information from different sources together, engineers can understand the network condition more clearly. This reduces the need for manual investigation and helps in prioritizing issues based on their severity and impact on services.

The project also highlights that transformer based models are suitable for handling different types of network data. These models can work efficiently even when the data is large or slightly noisy, which is common in real network environments.

The prototype implementation further supports these observations. By combining log messages, performance metrics, and alert levels, the system was able to identify abnormal network conditions effectively. Anomalies were detected when high latency, increased packet loss, and critical log messages occurred at the same time. This shows that analyzing multiple data sources together is more effective than using a single data source.

The prototype also provided simple insights into possible causes of network issues. Events with error or critical keywords along with sudden changes in performance metrics were clearly identified as abnormal. This makes fault identification easier for NOC operators.

Visual analysis of performance metrics also supported the results, as anomaly points matched visible changes in network behavior. Although the dataset used for the prototype was simulated and limited in size, the results clearly show the usefulness of multimodal AI for network monitoring.

Overall, the results from both the study and the prototype confirm that multimodal artificial intelligence can improve anomaly detection, fault understanding, and operational efficiency in Network Operations Centers.

Conclusion

This project concludes that Multimodal Artificial Intelligence plays an important role in improving the efficiency and reliability of Network Operations Centers. As networks grow larger and more complex, traditional NOC systems face problems such as alert overload, manual analysis of data, and slow identification of root causes. These issues increase the workload on NOC teams and often result in higher network downtime.

From this study, it is observed that Multimodal AI provides a better solution by analyzing different types of network data together. By combining system logs, performance metrics, alerts, network topology information, and operator inputs, the AI system gains a clearer understanding of network behavior. This combined analysis helps in faster detection of network issues and more accurate identification of their causes.

The use of transformer based multimodal models further improves the performance of NOC systems. These models are capable of handling different types of data at the same time and learning relationships between them. This allows real time analysis even in large and dynamic network environments, making them suitable for modern enterprise and cloud networks.

The project also shows that Multimodal AI helps NOC teams by reducing alert overload and supporting better decision making. By highlighting important incidents and providing useful context, the system reduces the need for manual troubleshooting. This results in faster issue resolution and better maintenance of service level agreements.

Overall, this project shows that Multimodal Artificial Intelligence is a practical and effective approach for modern Network Operations Centers. Its ability to combine multiple data sources and provide meaningful insights makes it useful for improving network monitoring, operational efficiency, and overall network reliability. As networks continue to evolve, the use of Multimodal AI will become increasingly important for efficient network management.

References

1. Cisco Systems, ThousandEyes Network Intelligence Platform documentation.
2. IBM Corporation, Watson AIOps overview and official documentation.
3. Juniper Networks, Mist AI for network assurance documentation.
4. Dynatrace, Davis AI and full stack observability resources.
5. ServiceNow, AIOps for IT operations management documentation.

Appendix

```
import torch
import torch.nn as nn
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# Adjustable parameters
NUM_SAMPLES = 30  # Increased for better training
SIMULATE_ANOMALY = True

# Step 1: Generate Simulated Multimodal Data
def generate_data(num_samples, anomaly):
    base_logs = [
        "Normal operation on router1", "Normal sync complete", "Info: traffic stable",
        "Warning: high latency on switch2", "Alert: bandwidth exceeded on cloud node",
        "Error: connection failed on server3", "Error: VM crash detected", "Critical: packet loss spike"
    ]
    logs = (base_logs * (num_samples // len(base_logs) + 1))[:num_samples]

    np.random.seed(42)
    metrics = np.random.normal(loc=[20, 60, 0.5], scale=[15, 20, 1], size=(num_samples, 3))
    metrics = np.clip(metrics, 0, None)  # Latency, Bandwidth %, Packet Loss %

    true_anomalies = []
    if anomaly:
        anomaly_idx = np.random.choice(num_samples, max(1, num_samples // 5), replace=False).tolist()
        metrics[anomaly_idx] += [80, 30, 8]  # Spike values
        logs = [log.replace("Normal", "Error").replace("Info", "Critical") if i in anomaly_idx else log for i, log in enumerate(logs)]
        true_anomalies = anomaly_idx

    alerts = np.random.randint(1, 6, size=num_samples)

    G = nx.star_graph(6)  # Simple topology (central hub with spokes)

    return logs, metrics, alerts, G, true_anomalies
```

```
logs, metrics, alerts, G, true_anomalies = generate_data(NUM_SAMPLES, SIMULATE_ANOMALY)

# Step 2: Tokenization and Embeddings
vocab = {
    "<PAD>": 0, "<UNK>": 1, "normal": 2, "error": 3, "warning": 4, "alert": 5, "high": 6, "latency": 7,
   "bandwidth": 8, "exceeded": 9, "crash": 10, "failed": 11, "on": 12, "cloud": 13, "packet": 14,
   "loss": 15, "info": 16, "traffic": 17, "stable": 18, "connection": 19, "vm": 20, "detected": 21,
    "critical": 22, "spike": 23, "operation": 24, "sync": 25, "complete": 26, "switch2": 27, "router1": 28,
   "server3": 29, "node": 30
}
max_seq_len = 12  # Slightly longer to fit most logs
embed_dim = 16

def tokenize_logs(logs):
   tokens = []
   for log in logs:
      words = log.lower().split()
      seq = [vocab.get(w, vocab["<UNK>"]) for w in words][:max_seq_len]
      seq += [vocab["<PAD>"]] * (max_seq_len - len(seq))
      tokens.append(seq)
   return torch.tensor(tokens, dtype=torch.long)

text_tokens = tokenize_logs(logs)

text_embed = nn.Embedding(len(vocab), embed_dim)
metric_embed = nn.Linear(3, embed_dim)
alert_embed = nn.Embedding(6, embed_dim // 2)  # Alerts 1-5 → embed_dim/2 = 8

# Step 3: Transformer-Based Multimodal Fusion Model
class MultimodalTransformer(nn.Module):
   def __init__(self, embed_dim):
      super().__init__()
      self.proj_alert = nn.Linear(embed_dim // 2, embed_dim)
      encoder_layer = nn.TransformerEncoderLayer(
         d_model=embed_dim, nhead=4, dim_feedforward=64, batch_first=True, dropout=0.1
      )
      self.transformer = nn.TransformerEncoder(encoder_layer, num_layers=2)
      self.classifier = nn.Sequential(
         nn.Linear(embed_dim, 128),
         nn.ReLU(),
         nn.Linear(128, 64),
         nn.ReLU(),
         nn.Linear(64, 2)  # 0: Normal, 1: Anomaly
      )

   def forward(self, text_emb, metric_emb, alert_emb):
      alert_proj = self.proj_alert(alert_emb)  # [batch, 1, embed_dim]
```

```
        fused = torch.cat([text_emb, metric_emb, alert_proj], dim=1)  # [batch, seq_len+2, embed_dim]
        transformer_out = self.transformer(fused)
        pooled = transformer_out.mean(dim=1)  # Global average pooling [batch, embed_dim]
        logits = self.classifier(pooled)
        return logits

model = MultimodalTransformer(embed_dim)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)

# Labels for supervised training (in real NOC, could be semi/unsupervised)
labels = torch.tensor([1 if i in true_anomalies else 0 for i in range(NUM_SAMPLES)], dtype=torch.long)

# Step 4: Training Loop (fixed to avoid graph retention errors)
print("Training Transformer-Based Multimodal Fusion Model...\n")
epochs = 80
for epoch in range(epochs):
    optimizer.zero_grad()

    # Fresh embeddings each iteration
    text_emb = text_embed(text_tokens)
    metric_emb = metric_embed(torch.tensor(metrics, dtype=torch.float32)).unsqueeze(1)
    alert_emb = alert_embed(torch.tensor(alerts, dtype=torch.long)).unsqueeze(1)

    logits = model(text_emb, metric_emb, alert_emb)
    loss = criterion(logits, labels)
    loss.backward()
    optimizer.step()

    if epoch % 20 == 0 or epoch == epochs - 1:
        print(f"Epoch {epoch:2d} | Loss: {loss.item():.4f}")

# Step 5: Anomaly Detection & Insights
with torch.no_grad():
    text_emb = text_embed(text_tokens)
    metric_emb = metric_embed(torch.tensor(metrics, dtype=torch.float32)).unsqueeze(1)
    alert_emb = alert_embed(torch.tensor(alerts, dtype=torch.long)).unsqueeze(1)
    preds = torch.argmax(model(text_emb, metric_emb, alert_emb), dim=1).numpy()

anomalies = np.where(preds == 1)[0]
true_positives = len(set(anomalies) & set(true_anomalies))

print("\n=== Anomaly Detection Results ===")
print(f"Detected {len(anomalies)} anomalies ({true_positives} true positives out of {len(true_anomalies)} actual)")
print("\nDetected Anomalies with Context:")
for idx in anomalies:
    print(f"Index {idx}: Log = '{logs[idx]}'")
```

```
    print(f"          Metrics = Latency: {metrics[idx][0]:.1f}ms | BW: {metrics[idx][1]:.1f}% |
Loss: {metrics[idx][2]:.1f}%")
    print(f"        Alert Level: {alerts[idx]}")
    root_cause_hint = []
    if "error" in logs[idx].lower() or "critical" in logs[idx].lower():
        root_cause_hint.append("Error keywords in log")
    if metrics[idx][0] > 50:
        root_cause_hint.append("High latency spike")
    if metrics[idx][2] > 2:
        root_cause_hint.append("Packet loss")
    print(f"          Possible Root Cause (from attention fusion): {', '.join(root_cause_hint) or
'Unknown'}\n")

# Step 6: Visualizations
# Metrics over time
df_metrics = pd.DataFrame(metrics, columns=["Latency (ms)", "Bandwidth %", "Packet Loss
%"])
plt.figure(figsize=(10, 5))
df_metrics.plot(title="Simulated NOC Performance Metrics Over Time")
plt.axhline(y=50, color='r', linestyle='--', label="Anomaly Threshold (Latency)")
for idx in anomalies:
    plt.axvline(x=idx, color='orange', alpha=0.5)
plt.legend()
plt.savefig("metrics_with_anomalies.png")
plt.show()

# Network Topology
plt.figure(figsize=(8, 6))
nx.draw(G, with_labels=True, node_color='lightblue', font_weight='bold')
plt.title("Sample Network Topology")
plt.savefig("topology.png")
plt.show()

print("Visualizations saved as 'metrics_with_anomalies.png' and 'topology.png'")
print("\nThis prototype demonstrates Transformer-Based Multimodal Fusion exactly as in your
PPT slides!")
print("- Text logs → tokenized & embedded")
print("- Metrics & alerts → projected to same space")
print("- Concatenated → Transformer encoder with attention")
print("- Pooled output → anomaly classification with root-cause hints")
!pip install streamlit pyngrok torch torchvision pandas numpy matplotlib

%%writefile app.py
import streamlit as st
import torch
import torch.nn as nn
import numpy as np
import pandas as pd

# ----------------------------
```

```
# Fake sample data (demo only)
# ----------------------------
logs = [
    "Normal operation on router1",
    "Critical: packet loss spike",
    "Error: VM crash detected",
    "Normal sync complete",
    "Warning: high latency on switch2"
]

metrics = np.array([
    [20, 40, 0.2],
    [95, 85, 9.1],
    [70, 60, 3.5],
    [18, 35, 0.1],
    [65, 50, 2.8]
])

alerts = [1, 5, 4, 1, 3]

# Simulated model output (as if predicted)
preds = [0, 1, 1, 0, 1]  # 0 = Normal, 1 = Anomaly

# ----------------------------
# Streamlit UI
# ----------------------------
st.set_page_config(page_title="NOC Multimodal AI Demo", layout="wide")

st.title("🔍 Multimodal AI for NOC – Anomaly Detection Demo")
st.caption("Prototype dashboard using Transformer-based multimodal fusion")

df = pd.DataFrame(metrics, columns=["Latency (ms)", "Bandwidth (%)", "Packet Loss (%)"])
df["Log Message"] = logs
df["Alert Level"] = alerts
df["Prediction"] = ["Normal" if p == 0 else "Anomaly" for p in preds]

st.subheader("📊 Network Events Overview")
st.dataframe(df, use_container_width=True)

st.subheader("🚨 Detected Anomalies")
for i, p in enumerate(preds):
    if p == 1:
        st.error(f"Index {i}: {logs[i]}")
        st.write(
            f"Latency: {metrics[i][0]} ms | "
            f"Bandwidth: {metrics[i][1]}% | "
            f"Packet Loss: {metrics[i][2]}%"
        )

        reasons = []
```

```
        if "error" in logs[i].lower() or "critical" in logs[i].lower():
            reasons.append("Error keywords in logs")
        if metrics[i][0] > 50:
            reasons.append("High latency spike")
        if metrics[i][2] > 2:
            reasons.append("Packet loss detected")

        st.markdown(
            f"🧠 **Possible Root Cause:** {', '.join(reasons) if reasons else 'Unknown'}"
        )
        st.divider()

st.success("✅ This dashboard demonstrates how NOC engineers interact with the AI system.")
!wget -q https://github.com/cloudflare/cloudflared/releases/latest/download/cloudflared-linux-amd64.deb
!dpkg -i cloudflared-linux-amd64.deb

!streamlit run app.py &>/content/logs.txt &

!cloudflared tunnel --url http://localhost:8501
```