## INFO8985 - In-Class Task 3 Report

Name: Jithin Jyothi  
Instructor: Prof. Richard  
Date: 2025-06-28

## Introduction:

In this assignment, we instrumented a monolithic Python Flask application using OpenTelemetry and SigNoz. The goal was to enable distributed tracing, structured logging, and metrics collection for better observability.

## Implementation Steps

1. Cloned and set up the Monolith Flask app in a Codespaces environment.

2. Added OpenTelemetry SDK and required Python libraries to instrument the code.

3. Modified the `app.py` to generate traces, logs, and custom metrics using OpenTelemetry.

4. Integrated SigNoz by cloning the SigNoz repo and running it via Docker Compose.

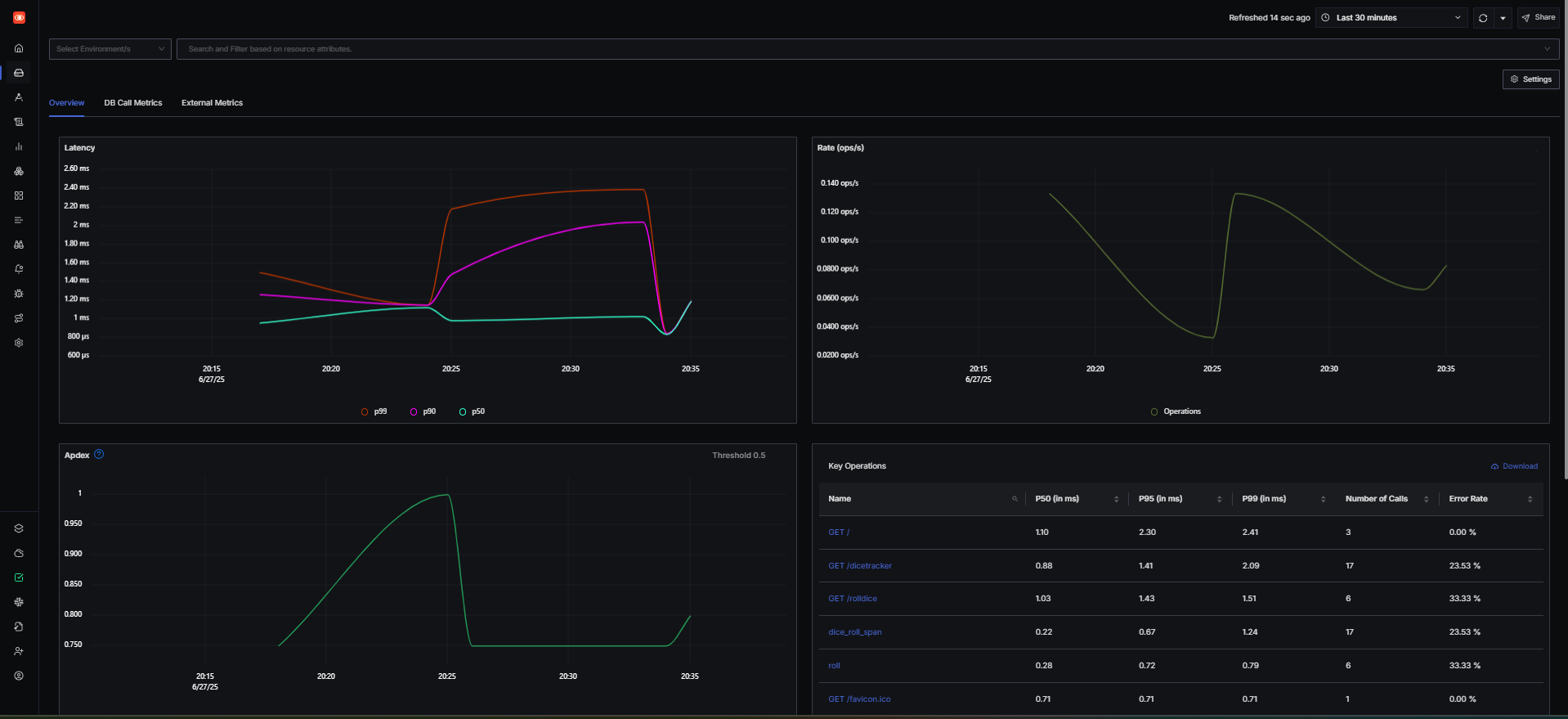
5. Created a custom OTLP collector config (`otel-collector-config.yaml`) to direct data to SigNoz.

6. Created a helper script (`run\_instrumented.sh`) to launch the app with instrumentation.

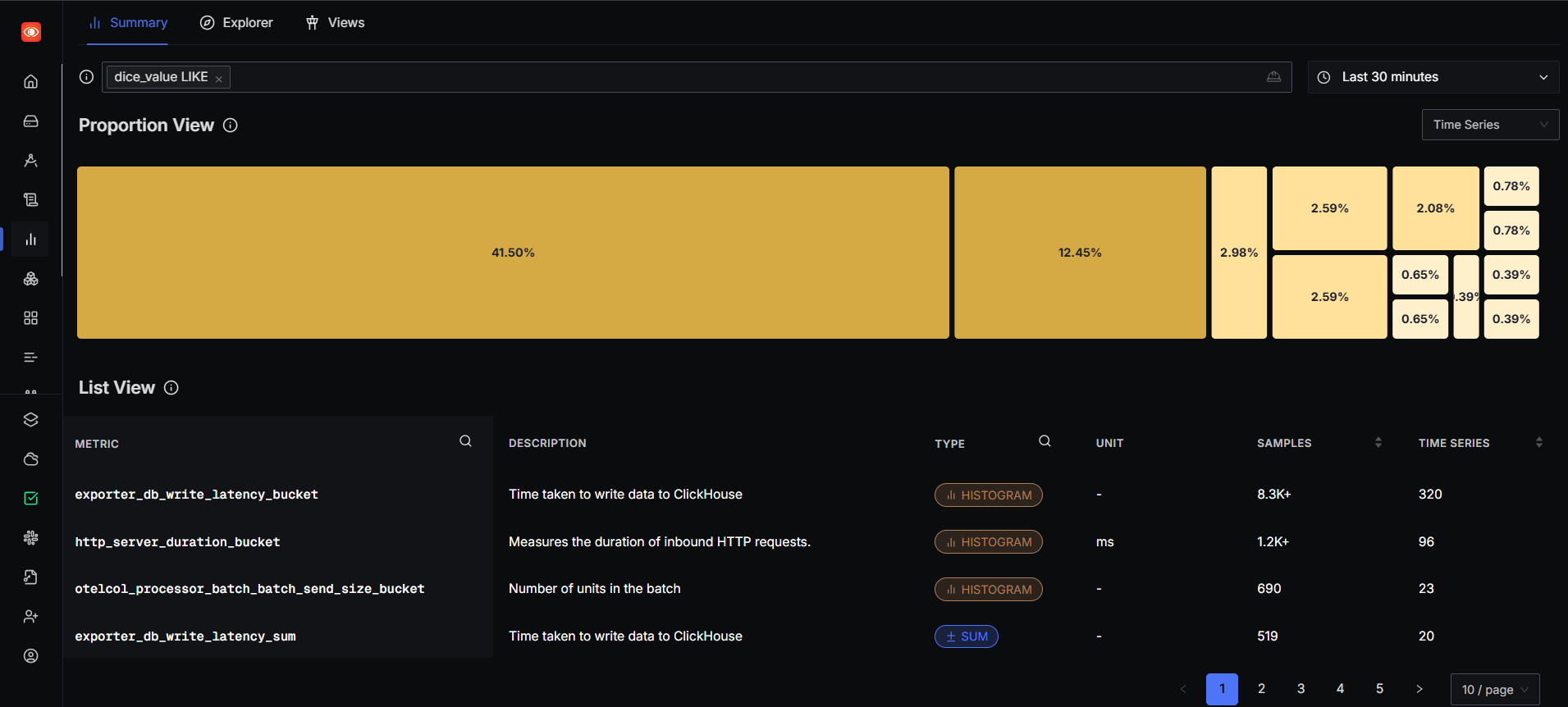
7. Verified metrics, logs, and traces in SigNoz UI.

## Screenshots & Output

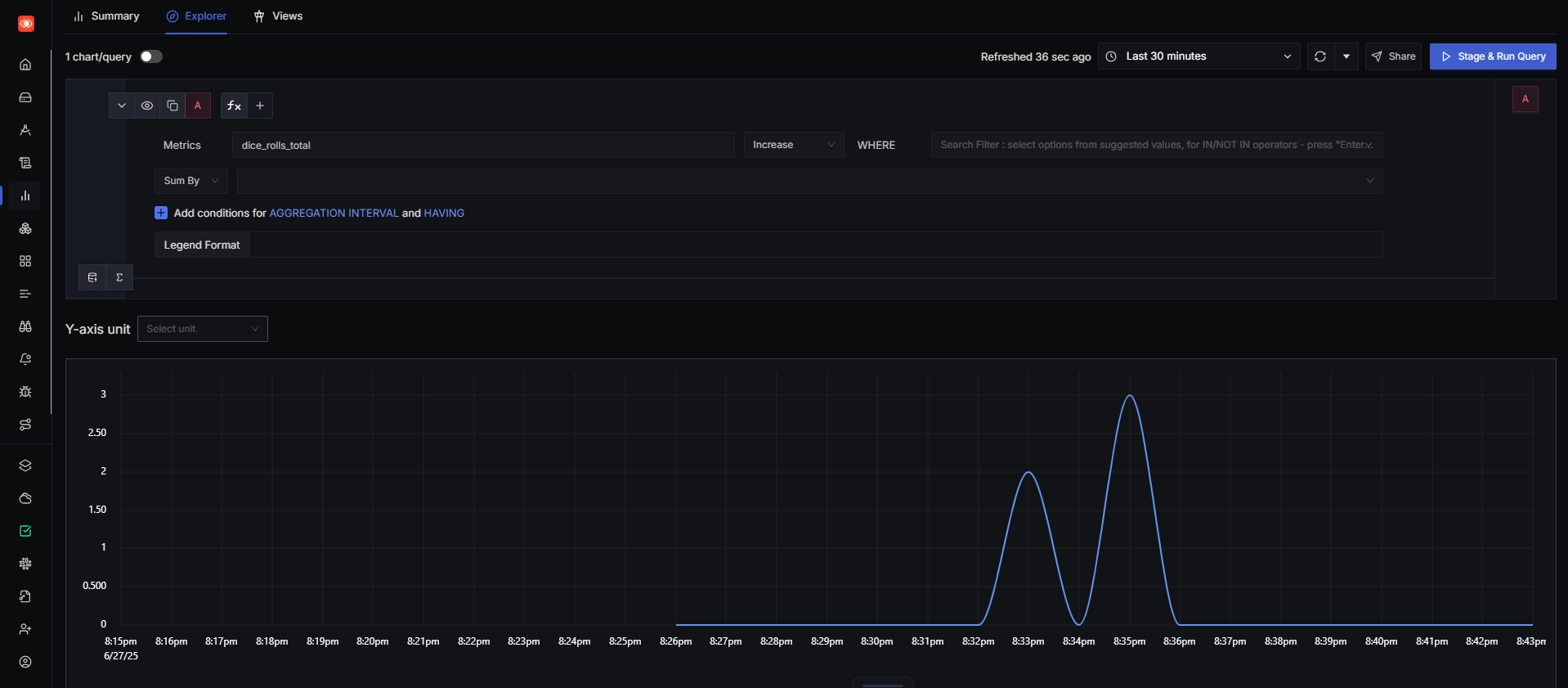
* SigNoz Services Overview - Shows the instrumented Flask service.



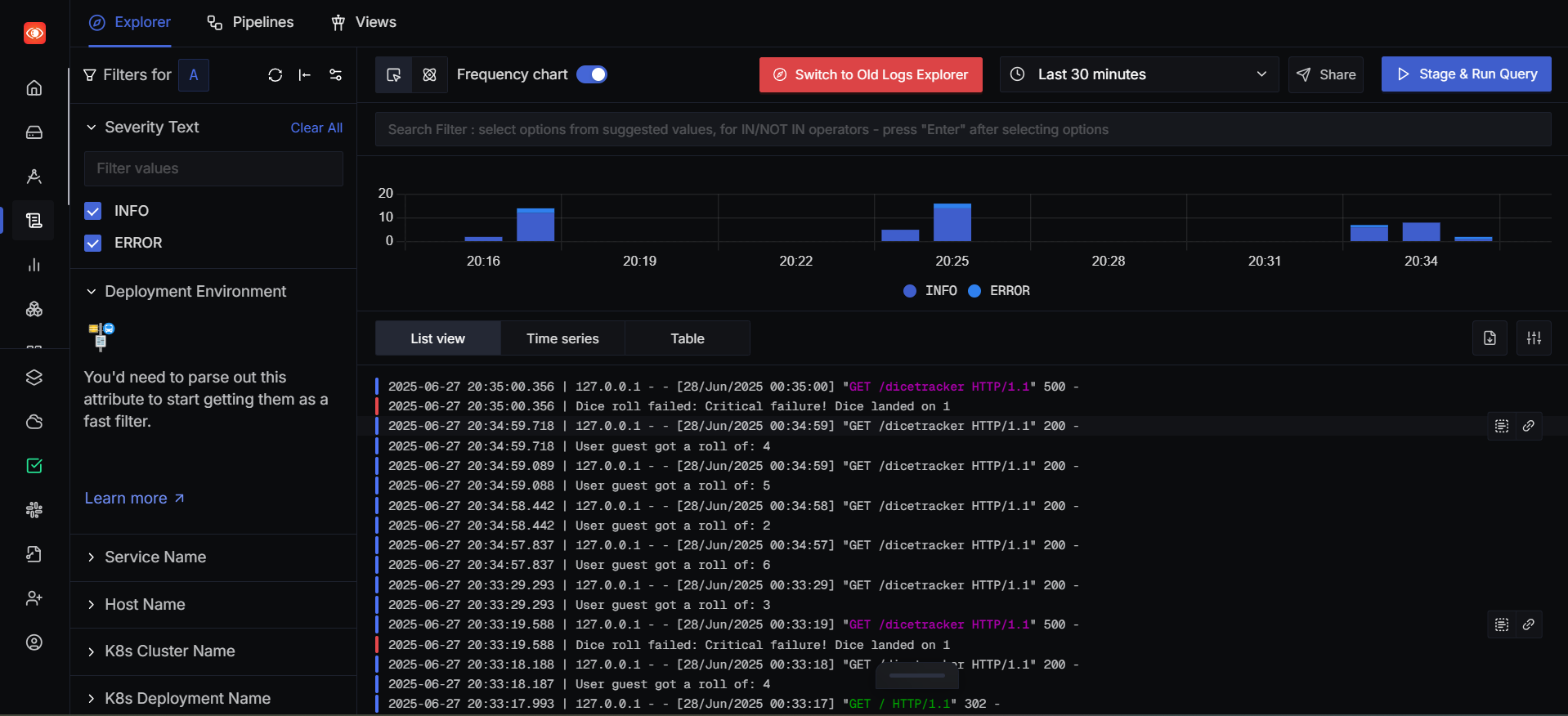
* Custom Metric Name "dice\_rolls" in SigNoz.



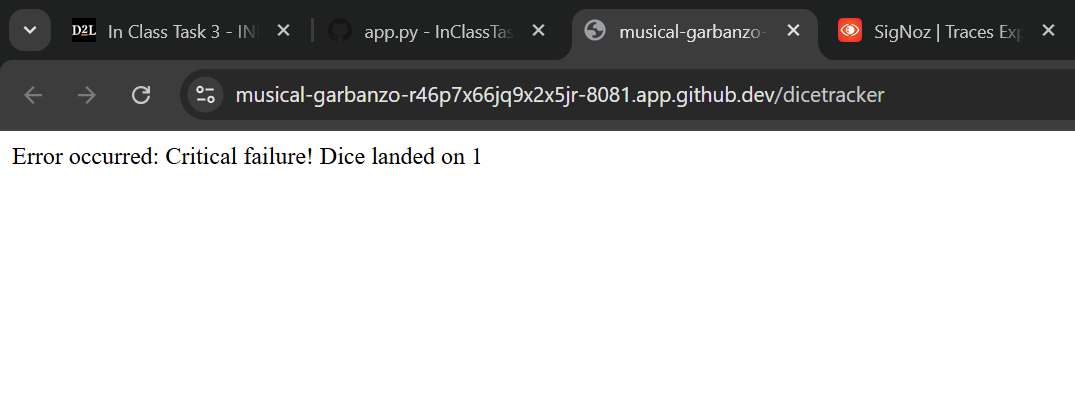
* Graph View of the custom "dice\_rolls\_total" metric.



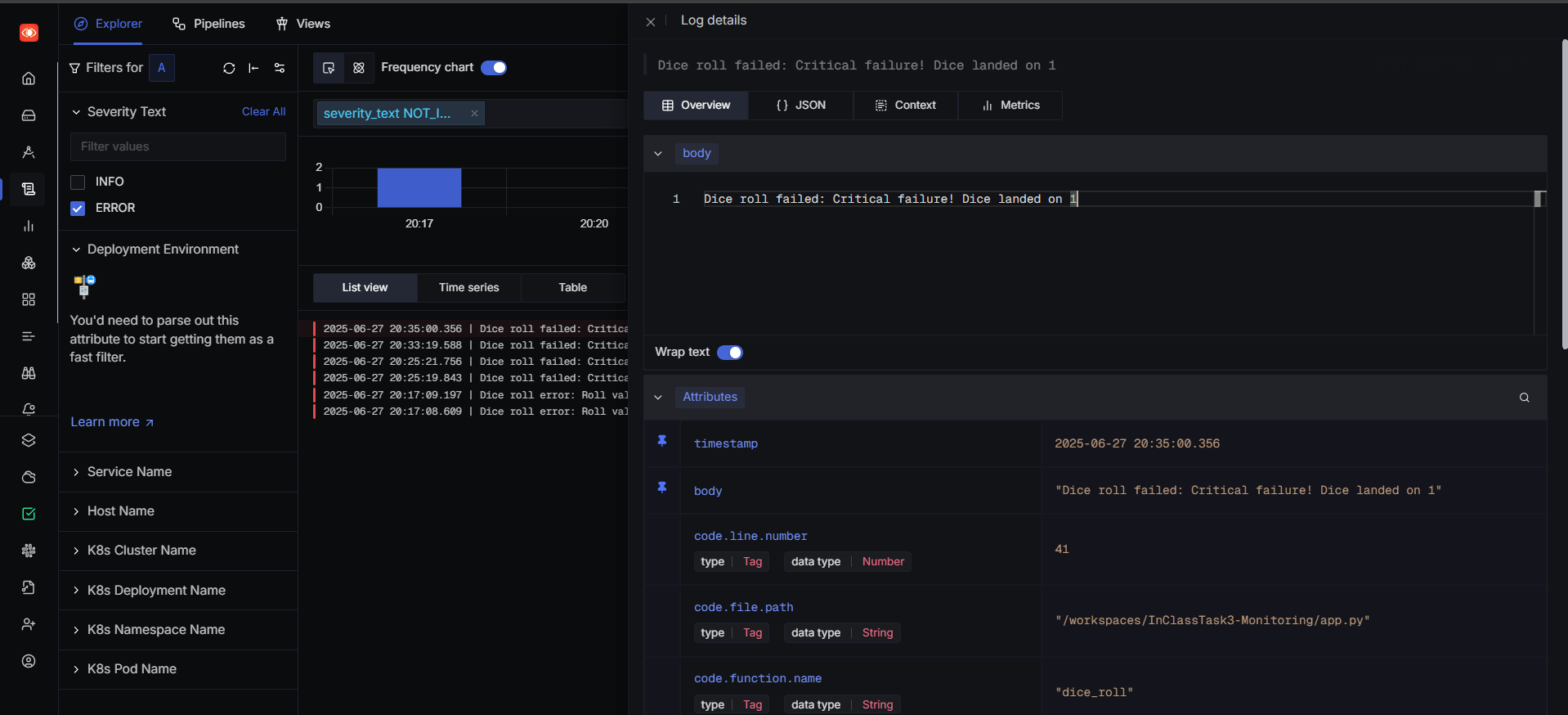
* Structured logs visible in SigNoz from Flask app.



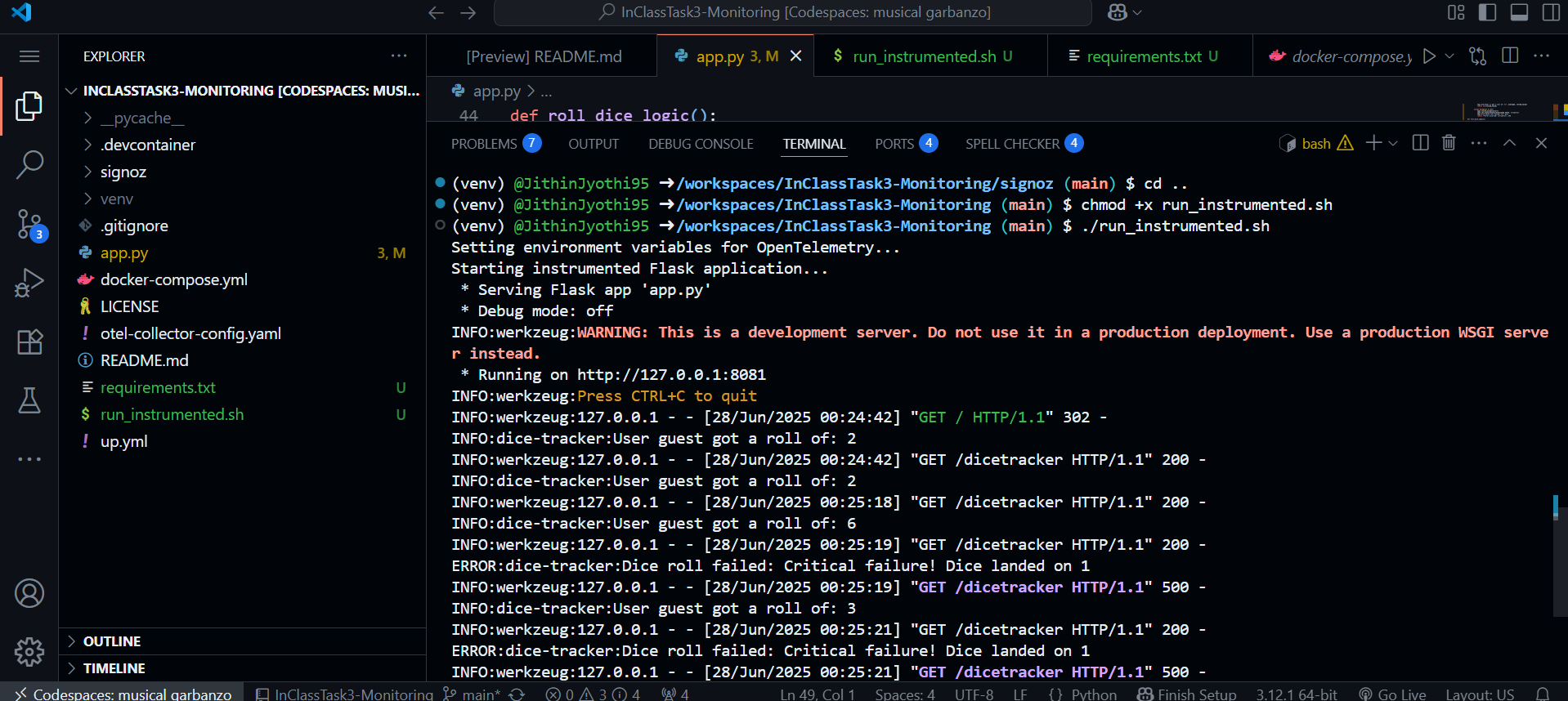
* Exception details captured and visualized in SigNoz.



* Error logs captured in SigNoz from application.



* Terminal output of the instrumented Flask app.



## Conclusion

The assignment demonstrated how to add observability to a Python application using modern tools like OpenTelemetry and SigNoz. By collecting metrics, traces, and logs, we were able to gain full visibility into the application's behavior and performance.