# EIRO Platform

**Enterprise Incident Response Orchestrator**

**Repository**: [https://github.com/ronitbhatia/EIRO-Kaggle-Capestone](https://github.com/ronitbhatia/EIRO-Kaggle-Capestone)

## Track

**Enterprise Agents**

## Problem Statement

Enterprise IT operations teams face constant pressure to quickly identify, investigate, and resolve incidents that impact business operations. Traditional incident management processes are often manual, slow, and lack intelligent coordination between different stages of incident response. When an incident occurs, multiple teams need to be involved: triage specialists classify the issue, investigation teams analyze root causes, resolution teams implement fixes, and communication teams keep stakeholders informed. This fragmented approach leads to:

- **Delayed response times**: Manual handoffs between teams create bottlenecks

- **Inconsistent prioritization**: Human judgment varies, leading to misprioritized incidents

- **Knowledge gaps**: Investigators may not have access to relevant documentation or historical solutions

- **Poor visibility**: Stakeholders lack real-time updates on incident status

- **Limited learning**: No systematic evaluation of how well incidents were handled

## Solution Overview

**EIRO Platform** (Enterprise Incident Response Orchestrator) is a multi-agent AI system that automates and orchestrates the entire incident lifecycle from initial report to resolution. The system uses four specialized agents working in sequence:

**Triage Agent**: Automatically classifies incidents, assigns priority, and categorizes them

**Investigation Agent**: Analyzes root causes using diagnostic tools and knowledge base searches

**Resolution Agent**: Generates step-by-step resolution plans and documents fixes

**Communication Agent**: Keeps stakeholders informed at each stage

A fifth **Evaluation Agent** (LLM-as-judge) assesses the performance of each agent to ensure quality and continuous improvement.

# Architecture

## Multi-Agent System Design

EIRO Platform implements a **sequential orchestration pattern** where agents work in a defined workflow:

```
    Incident Report → Triage Agent → Investigation Agent → Resolution Agent → Communication Agent
                                                                        ↓
                                                              Evaluation Agent (LLM-as-Judge)
```

**Key Design Decisions:**

• **Sequential Flow**: Each agent completes its task before the next begins, ensuring proper context accumulation

• **State Management**: Session manager tracks incident state (triage → investigation → resolution → closed)

• **Tool Integration**: Each agent has access to relevant tools (database, diagnostics, knowledge base, notifications)

• **Observability**: All agent actions are logged and traced for debugging and analysis

# Agent Responsibilities

## 1. Triage Agent

- **Input**: Raw incident report (title, description, severity, reporter)

- **Process**: Analyzes incident using LLM to extract priority, category, and initial assessment

- **Output**: Classified incident with priority (low/medium/high/critical) and category (performance/error/connectivity/security/other)

- **Tools Used**: Incident database (update incident record)

## 2. Investigation Agent

- **Input**: Classified incident from Triage Agent

- **Process**:

- Runs system diagnostic tools to check component health

- Performs symptom-based diagnosis

- Searches knowledge base for similar incidents and solutions

- Synthesizes findings into root cause analysis

- **Output**: Root cause analysis with evidence and recommended resolution approach

- **Tools Used**: System diagnostics, knowledge base search, incident database

## 3. Resolution Agent

- **Input**: Root cause analysis from Investigation Agent

- **Process**: Generates detailed resolution plan with step-by-step actions, verification steps, and prevention measures

- **Output**: Resolution plan and summary for documentation

- **Tools Used**: Incident database (update with resolution)

## 4. Communication Agent

- **Input**: Current incident state and stage
- **Process**: Generates stakeholder-friendly status updates at each stage
- **Output**: Notifications sent to incident reporter
- **Tools Used**: Notification service

### 5. Evaluation Agent (LLM-as-Judge)

- **Input**: Agent responses and context
- **Process**: Evaluates each agent's performance on accuracy, completeness, clarity, actionability, and efficiency
- **Output**: Scores (1-10) and recommendations (excellent/good/needs_improvement/poor)

# Tools and Agent Capabilities

## Custom Tools Implemented

EIRO Platform includes four custom tools that demonstrate the **Tools** concept from the course:

### 1. Incident Database Tool (`tools/incident_db.py`)

- **Purpose**: CRUD operations for incident records
- **Functions**:
- `create_incident_tool()`: Create new incident records
- `get_incident_tool()`: Retrieve incident by ID
- `update_incident_tool()`: Update incident fields
- `list_incidents_tool()`: Query incidents with filters
- **Usage**: All agents use this to read and update incident state

### 2. System Diagnostic Tool (`tools/system_diagnostics.py`)

- **Purpose**: Simulate system health checks and diagnostics
- **Functions**:
- `check_system_health()`: Check component health (database, API server, cache, etc.)
- `diagnose_issue()`: Symptom-based diagnosis with likely causes
- `get_system_metrics()`: Detailed metrics for components
- **Usage**: Investigation Agent uses this to gather diagnostic data

### 3. Knowledge Base Tool (`tools/knowledge_base.py`)

- **Purpose**: Search internal documentation and historical solutions
- **Functions**:
- `search_knowledge_base()`: Semantic search with relevance scoring
- `get_article()`: Retrieve specific knowledge base articles
- `get_articles_by_category()`: Filter by category
- **Usage**: Investigation Agent searches for similar incidents and solutions

### 4. Notification Tool (`tools/notifications.py`)

- **Purpose**: Send alerts and status updates to stakeholders
- **Functions**:
- `send_notification_tool()`: Send notifications with priority levels
- **Usage**: Communication Agent sends updates at each stage

## Tool Integration Pattern

Tools are integrated as Python functions that agents can call. The orchestrator provides tool context to agents through their prompts, and agents reference tool outputs in their reasoning. This demonstrates how custom tools extend agent capabilities beyond pure LLM reasoning.

# Memory and Context Strategy

## Session Management

EIRO Platform implements **Sessions and Memory** through a `SessionManager` class that maintains persistent state for each incident:

- **Session Creation**: Created when incident is first reported
- **State Tracking**: Tracks incident lifecycle state (triage → investigation → resolution → closed)
- **History Logging**: Records all agent actions and decisions with timestamps
- **Context Accumulation**: Each agent's output becomes context for the next agent

**Key Features:**

- Session data persists across agent interactions
- Full history available for debugging and audit
- State transitions are explicit and tracked

## Context Engineering

- **Progressive Context Building**: Each agent receives accumulated context from previous agents
- **Selective Information**: Only relevant information is passed to each agent (e.g., Investigation Agent gets triage results, not raw incident)
- **Tool Results Integration**: Diagnostic and knowledge base results are included in agent prompts

# Observability and Evaluation

## Observability Implementation

EIRO Platform includes comprehensive observability through the `ObservabilityLogger` class:

- **Structured Logging**: All agent actions logged with level, agent name, message, and context

- **Distributed Tracing**: Traces track operations across multiple agents with spans

- **Metrics**: Tracks agent calls, tool calls, and errors

- **Real-time Visibility**: Logs printed to console for immediate feedback (useful in Kaggle notebooks)

**Example Trace Structure:**

```
Trace: incident-1
  Span: triage (TriageAgent, 2.3s)
  Span: investigation (InvestigationAgent, 5.1s)
  Span: resolution (ResolutionAgent, 3.7s)
  Total Duration: 11.1s
```

## Agent Evaluation

EIRO Platform implements **Agent Evaluation** using **LLM-as-Judge** methodology:

- **Evaluation Criteria**:

- Accuracy (1-10): Correctness of response

- Completeness (1-10): Coverage of all aspects

- Clarity (1-10): Communication quality

- Actionability (1-10): Practical value

- Efficiency (1-10): Conciseness

- **Evaluation Process**:

Each agent's response is evaluated after completion

LLM judge (Gemini) analyzes the response against criteria

Scores and recommendations are generated

Results stored for analysis and improvement

- **Benefits**:
- Continuous quality monitoring
- Identifies agents needing improvement
- Provides feedback for prompt engineering
- Enables A/B testing of agent strategies

# Example User Flows

## Flow 1: High-Priority Performance Incident

```
1. User reports: "Database connection timeouts, 40% error rate increase"
2. Triage Agent:
   - Classifies as Priority: High, Category: Performance
   - Updates incident record
3. Investigation Agent:
   - Checks system health (database: degraded)
   - Diagnoses: "Performance degradation - likely database query latency"
   - Searches KB: Finds "Database Connection Timeout Resolution" article
   - Root cause: Connection pool exhaustion
4. Resolution Agent:
   - Plan: Increase connection pool size, add read replicas
   - Verification: Monitor connection metrics
   - Prevention: Auto-scaling connection pools
5. Communication Agent:
   - Sends update: "Incident triaged and root cause identified. Resolution in progress."
6. Evaluation Agent:
   - Triage: 8.5/10 (excellent)
   - Investigation: 9.0/10 (excellent)
   - Resolution: 8.0/10 (good)
```

## Flow 2: Low-Priority Connectivity Issue

```
1. User reports: "Cannot connect to file storage"
2. Triage Agent: Priority: Low, Category: Connectivity
3. Investigation Agent:
   - System health: File storage component healthy
   - Diagnosis: "Connectivity issue - likely permission problem"
   - KB search: Finds "File Storage Access Denied" article
4. Resolution Agent:
   - Plan: Check IAM roles and service account credentials
5. Communication Agent: Sends status update
```

```
    6. Evaluation: All agents score 7.5-8.5/10
```

# Impact and Limitations

## Impact

- **Faster Resolution**: Automated orchestration reduces manual handoffs and delays

- **Consistent Quality**: Standardized agent behavior ensures consistent triage and investigation

- **Knowledge Utilization**: Knowledge base integration helps leverage organizational knowledge

- **Visibility**: Communication agent keeps stakeholders informed automatically

- **Continuous Improvement**: Evaluation agent provides feedback for system refinement

## Limitations

- **Simulated Environment**: Diagnostic tools and knowledge base are simulated (not production systems)

- **LLM Dependency**: All reasoning depends on LLM quality; may hallucinate or make errors

- **No Actual Fix Execution**: Resolution Agent generates plans but doesn't execute them (safety consideration)

- **Single-Tenant**: Current implementation handles one incident at a time

- **Evaluation Overhead**: LLM-as-judge adds latency and cost

## Future Work

**Parallel Processing**: Handle multiple incidents concurrently

**Actual Tool Integration**: Connect to real monitoring systems (Datadog, New Relic, etc.)

**Human-in-the-Loop**: Add approval gates for critical incidents

**Learning System**: Use evaluation feedback to improve agent prompts

**Long-Running Operations**: Support incidents that require pause/resume

**A2A Protocol**: Enable agents to communicate directly via A2A

**Deployment**: Deploy to Vertex AI Agent Engine for production use

**Advanced Context Engineering**: Implement context compaction for long conversations

# Course Concepts Demonstrated

This project demonstrates three required concepts:

**Multi-Agent System**: Sequential orchestration of four specialized agents (Triage, Investigation, Resolution, Communication) with clear handoffs and state management

**Tools**: Four custom tools (Incident Database, System Diagnostics, Knowledge Base, Notifications) that extend agent capabilities beyond pure LLM reasoning

**Agent Evaluation**: LLM-as-judge evaluation system that assesses agent performance on multiple criteria and provides actionable feedback

Additionally, the project includes:

- **Sessions and Memory**: Session manager maintains state and history across agent interactions
- **Observability**: Comprehensive logging, tracing, and metrics

# Technical Implementation

## Technology Stack

- **LLM**: Google Gemini 2.5 Flash (via `google-generativeai`)

- **Language**: Python 3.x
- **Architecture**: Modular agent system with tool integration
- **Deployment**: Kaggle Notebook compatible

### Key Files

- `main.py`: Main orchestration and agent definitions
- `tools/`: Custom tool implementations
- `utils/`: Session management and observability
- `evaluation/`: LLM-as-judge evaluation

### Running the System

```
from main import IncidentOrchestrator

orchestrator = IncidentOrchestrator(api_key="your-api-key")

result = orchestrator.handle_incident(
    title="Database Connection Timeout",
    description="Users experiencing timeouts...",
    reporter="ops@company.com",
    severity="high",
    evaluate=True
)
```

## Conclusion

EIRO Platform demonstrates how multi-agent AI systems can automate complex enterprise workflows. By combining specialized agents, custom tools, session management, observability, and evaluation, EIRO Platform provides a production-ready pattern for intelligent incident management. The system showcases the power of agent orchestration while maintaining transparency through logging and evaluation.