

```
# Log details
print(f"[Cost] {agent_type} LLM call: ${cost_usd:.4f} "
      f"({usage.get('input_tokens', 0)} in, {usage.get('output_tokens', 0)} out)")

def record_tool_call(
    self,
    agent_type: str,
    tool_name: str,
    duration_seconds: float
):
    """Record tool invocation (typically free but track for metrics)"""
    self.investigation.tool_calls += 1

    # Track tool usage
    key = f"tool_{tool_name}"
    self.cost_breakdown[key] = self.cost_breakdown.get(key, 0) + 1

    print(f"[Tool] {agent_type} called {tool_name} ({duration_seconds:.2f}s)")

def check_budget(self, max_cost_usd: float) -> bool:
    """Check if investigation is within budget"""
    return self.investigation.total_cost_usd < max_cost_usd

def get_cost_report(self) -> Dict:
    """Generate cost breakdown report"""
    return {
        "total_cost_usd": self.investigation.total_cost_usd,
        "llm_calls": self.investigation.llm_calls,
        "tool_calls": self.investigation.tool_calls,
        "duration_minutes": self.investigation.duration_minutes,
        "cost_per_minute": (
            self.investigation.total_cost_usd / self.investigation.duration_minutes
            if self.investigation.duration_minutes > 0 else 0
        ),
        "breakdown": self.cost_breakdown
    }
```

please proceed with Part 4, following all instructions as before.



Incident investigation MCP application

Sonnet 4.5