# Opik Integration in FretCoach

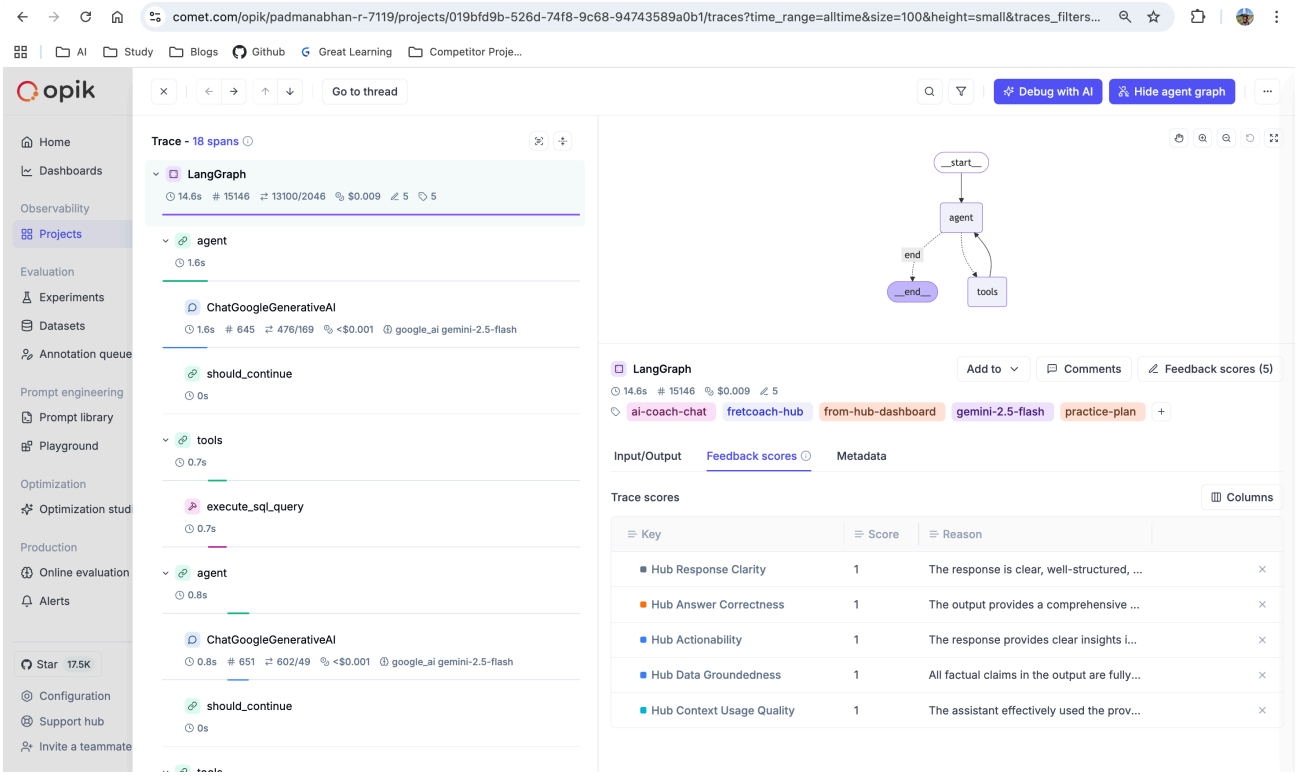
## Features Implemented

### 1. Traces with Metadata and Tags

All LLM calls are logged as traces in Opik with proper tags to organize and filter different types of interactions.

**Hub Coach Chats:**

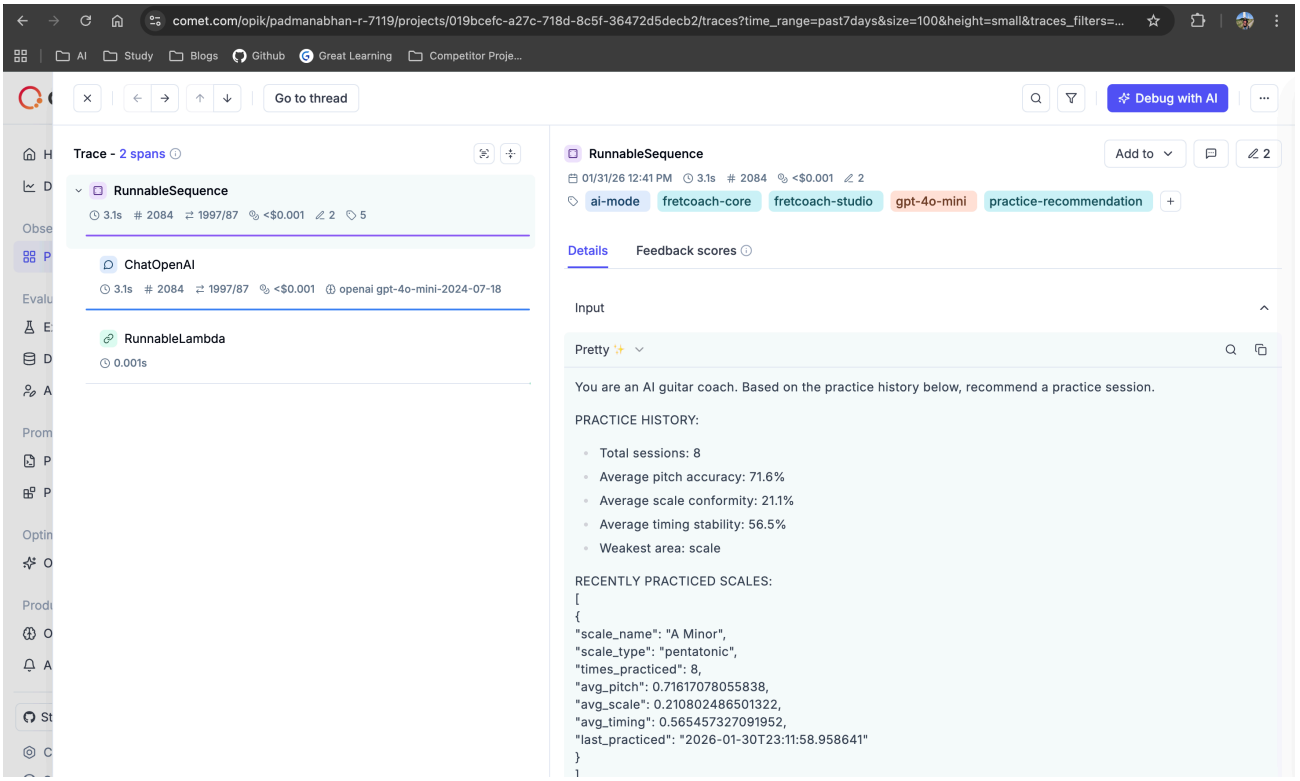
- Tags: `ai-coach-chat`, `fretcoach-hub`, `from-hub-dashboard`, `gemini-2.5-flash`, `practice-plan`
- Used for AI coach conversations in the web dashboard




*Hub coach chat traces with proper tags in Opik dashboard*

**AI Mode (Practice Recommendations):**

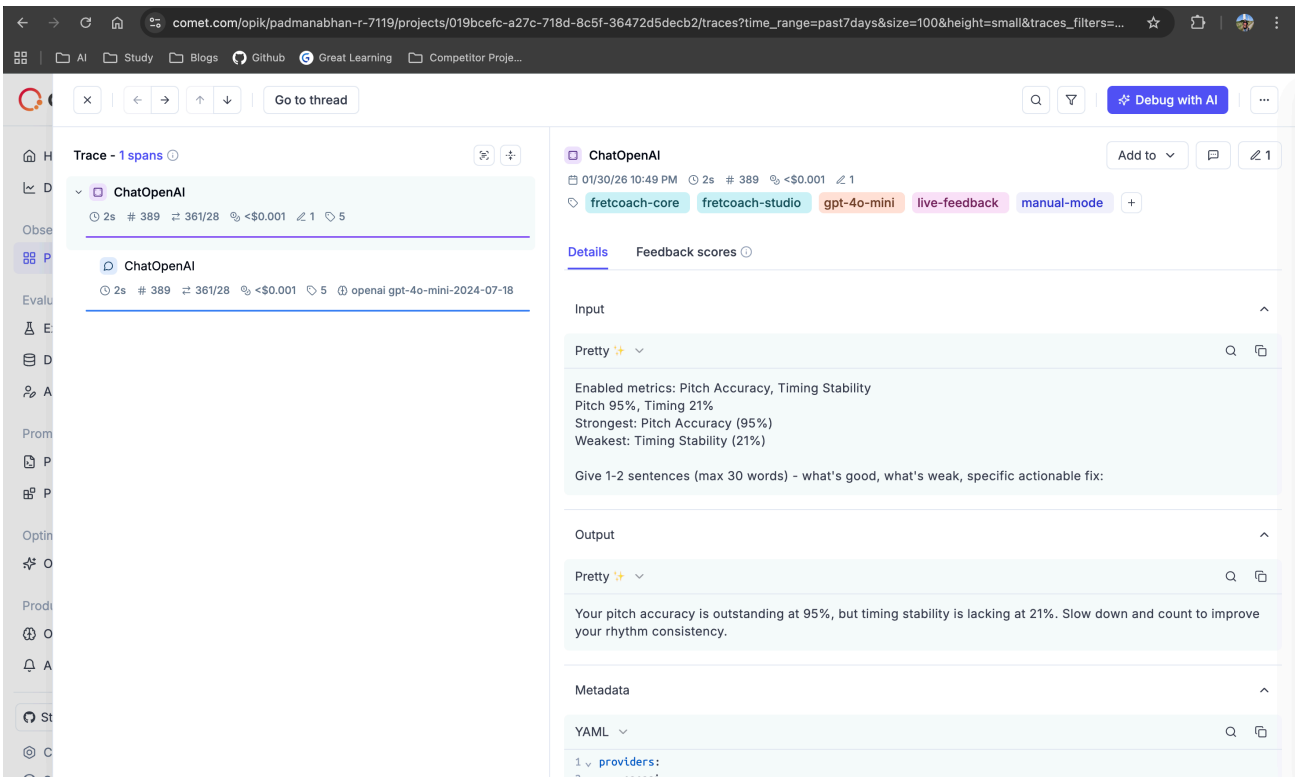
- Tags: `fretcoach-core`, `gpt-4o-mini`, `ai-mode`, `fretcoach-studio`, `practice-recommendation`
- Used for generating personalized practice recommendations




*AI mode practice recommendation traces*

**Live AI Feedback in Session:**

- Tags: `fretcoach-core` , `gpt-4o-mini` , `ai-mode` , `fretcoach-studio` , `live-feedback`
- Used for real-time coaching feedback during practice sessions




*Live AI feedback traces during practice sessions*

## 2. Thread Management

Each trace uses a structured `thread_id` to group related LLM calls and maintain conversation context across multiple requests.

**Thread Naming Conventions:**

**Hub Coach Chats:**

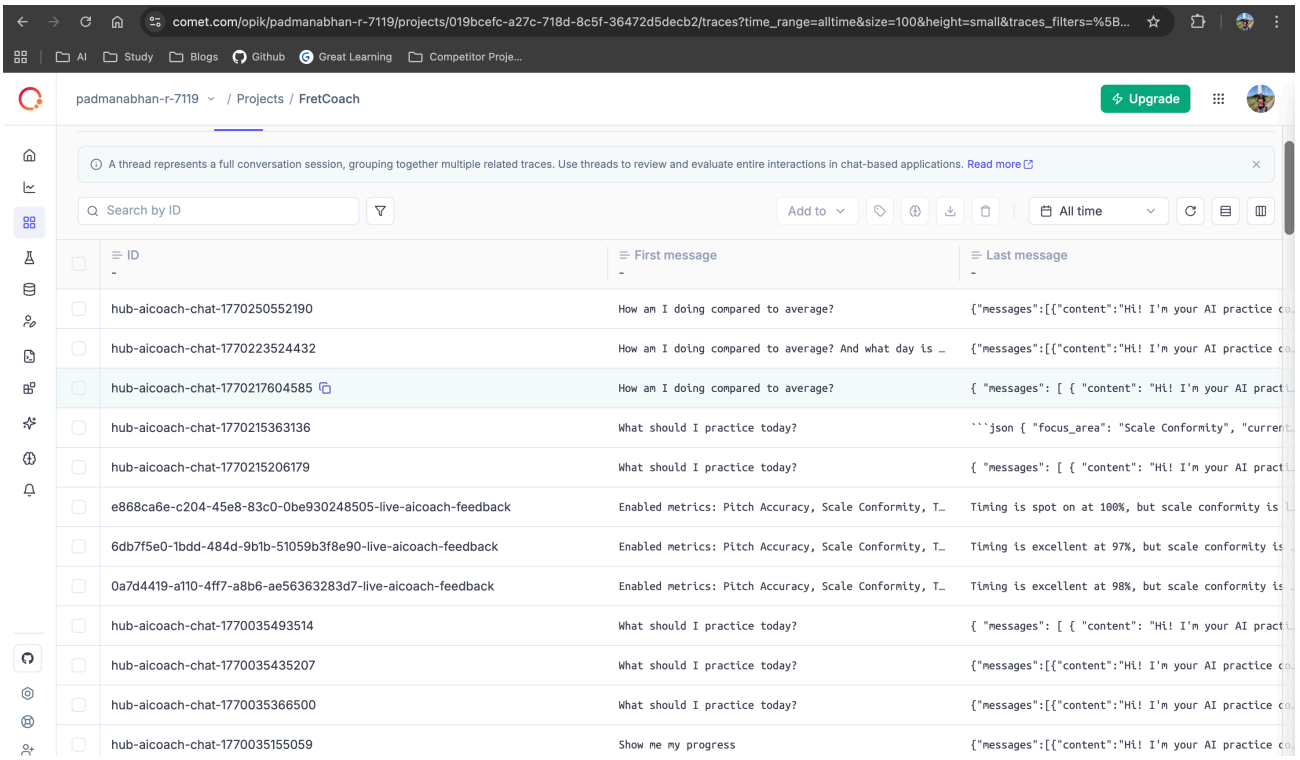
- Format: `hub-{user_id}`
- Example: `hub-default_user`
- Groups all coach chat messages for a user's conversation session

**AI Mode (Practice Recommendations):**

- Format: `{deployment}-ai-mode-{practice_id}`
- Example: `studio-ai-mode-a1b2c3d4-e5f6-7890-abcd-ef1234567890`
- Maintains thread across multiple recommendation requests until session starts
- Uses practice_id to persist thread even when user requests new recommendations

**Live AI Feedback:**

- Format: `{session_id}-live-aicoach-feedback`
- Example: `abc123-live-aicoach-feedback`
- Groups all live feedback calls within a single practice session



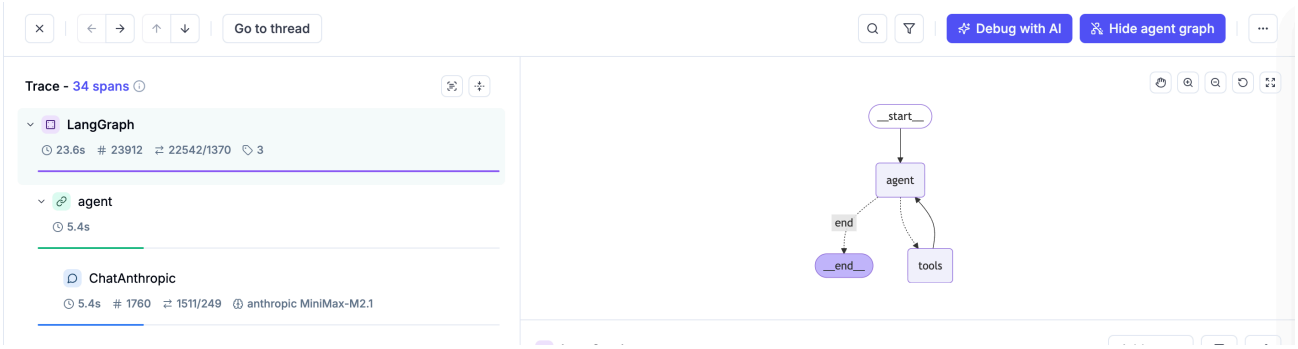

*Thread IDs grouping related traces in Opik*

---

## 3. Agent Graph Visualization

LangGraph execution flows visualized in Opik for hub coach chats using `workflow.get_graph(xray=True)`.

Shows agent reasoning path: agent → tool calls ( `execute_sql_query` , `get_database_schema` ) → decision nodes → response.




*LangGraph agent execution flow in Opik*

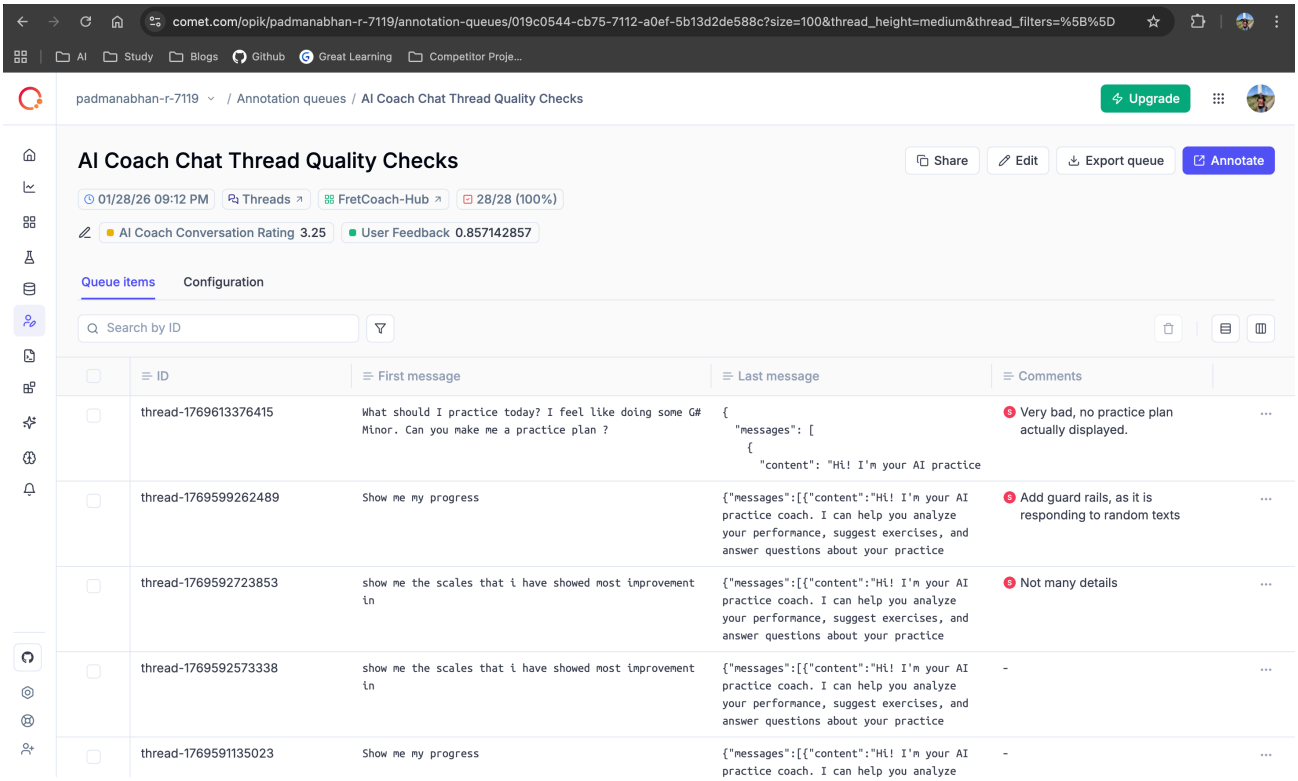
---

## 4. Annotation Queues

Used annotation queues to review and annotate agent outputs for quality evaluation.

**Implementation:**

- Created custom feedback definitions for LLM output quality
- Reviewed agent responses in annotation queues
- Rated outputs using custom criteria
- Human-in-the-loop evaluation for agent improvements




*Annotation queue for with reviewed LLM outputs*

---

## 5. Datasets and Prompts

Created and saved datasets and prompts for use in experiments, evaluations, and playground testing.

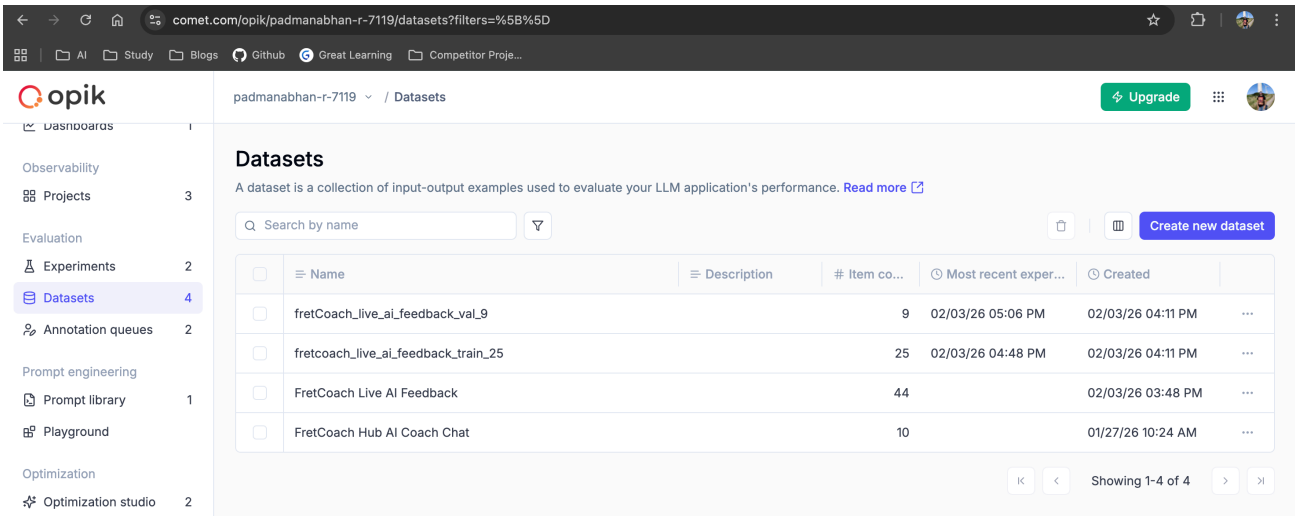
**Datasets:**

- Curated test cases from real user sessions

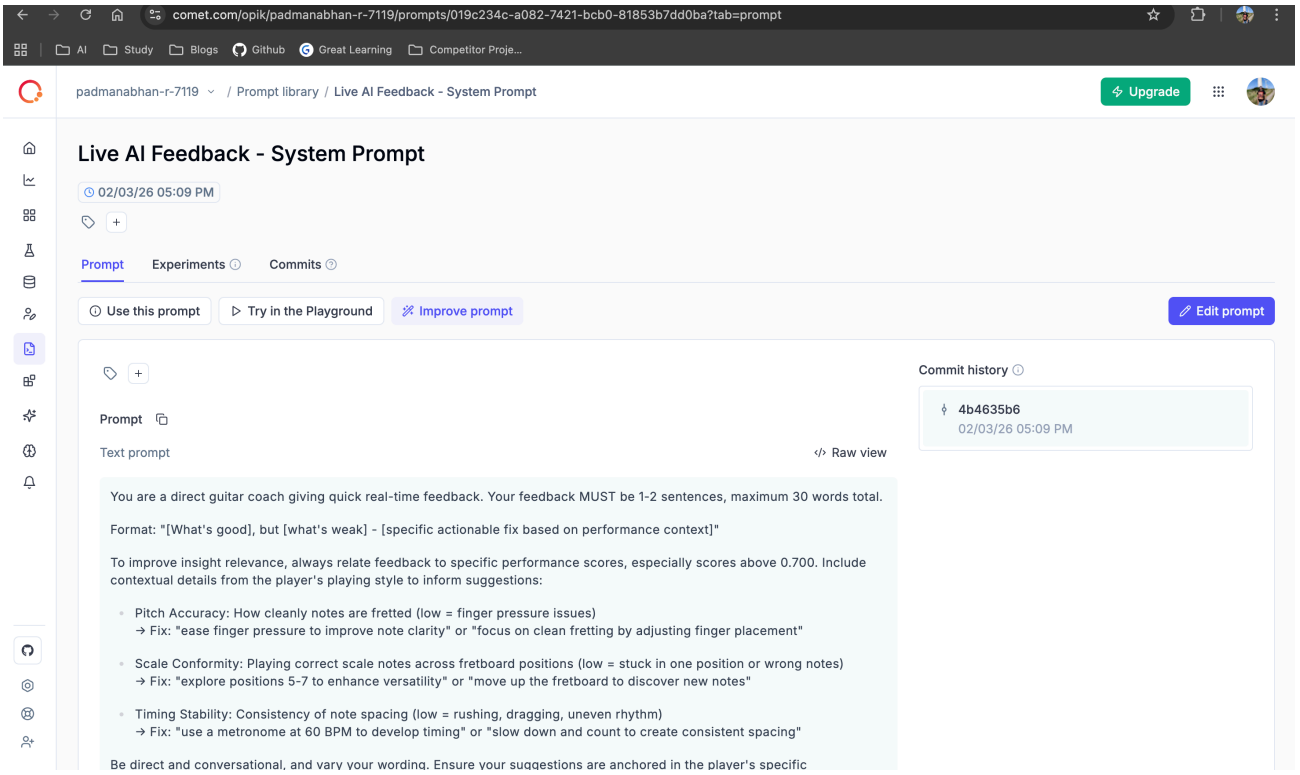
- Saved to Opik for reproducible evaluations
- Used across experiment runs for consistent testing

**Prompts:**

- Stored coaching prompt templates
- Version controlled in Opik
- Used in playground for rapid iteration




*Datasets created for experiment runs*




*Saved prompts*

---

## 6. Experiments and Custom Metrics

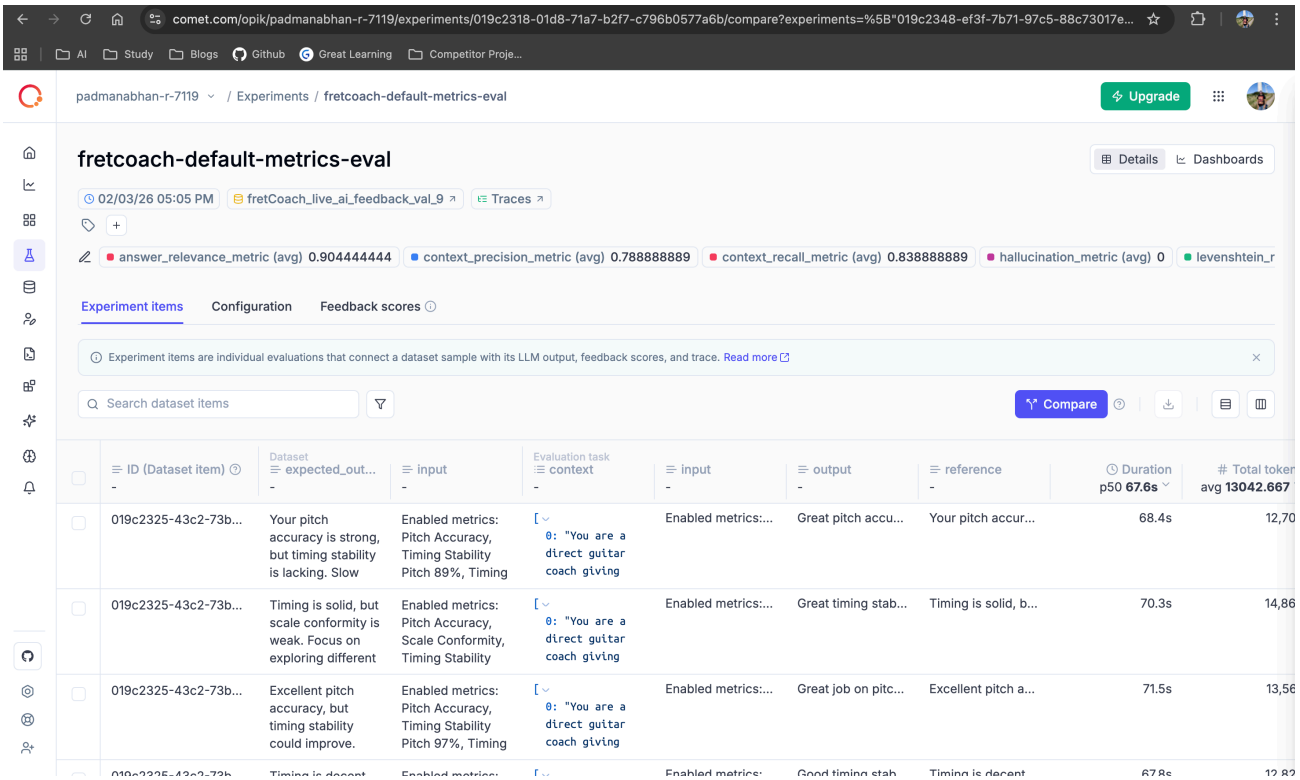
Ran experiments to evaluate LLM performance using both default Opik metrics and custom-created metrics.

**Default Metrics:**

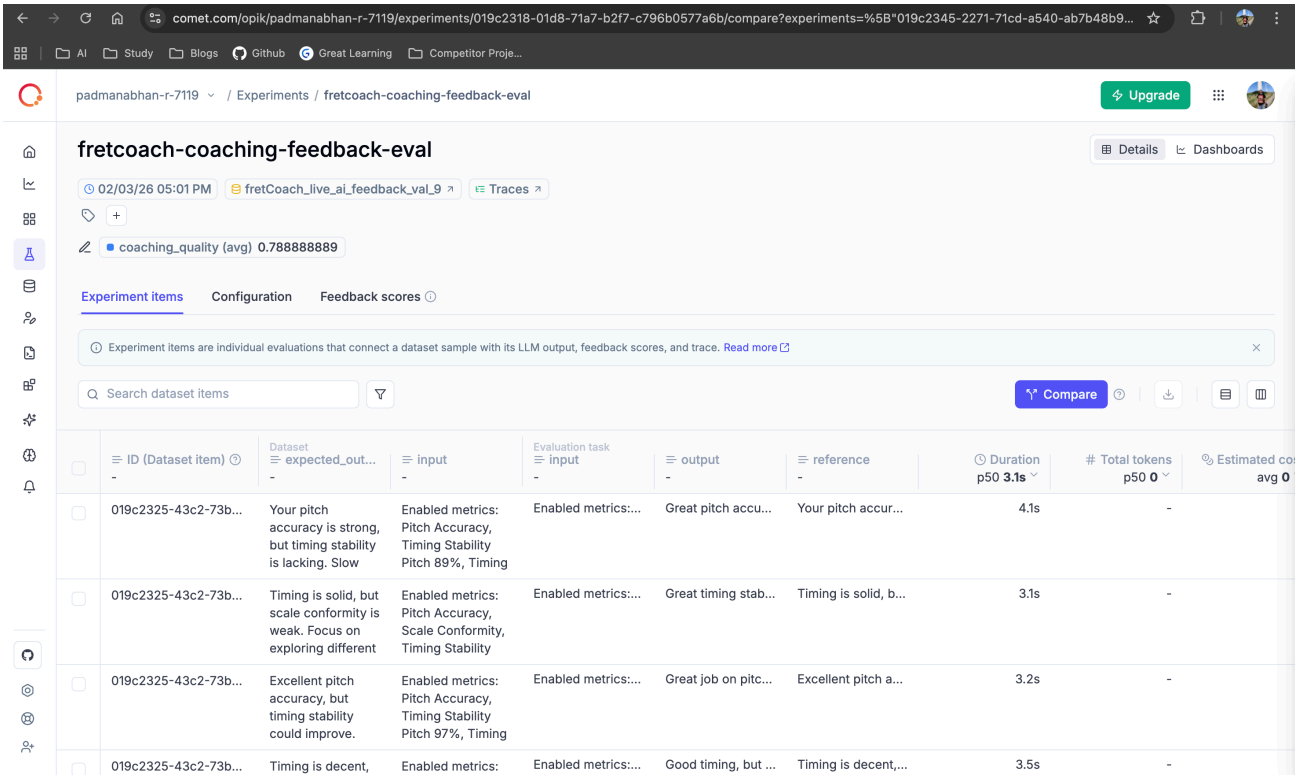
- Used Opik's built-in evaluation metrics
- Measured response quality across test datasets

**Custom Metrics:**

- Created custom metric for domain-specific evaluation
- Tailored to guitar coaching context
- Measured coaching quality and relevance




*Experiments with default Opik metrics*




*Experiments with custom metric*

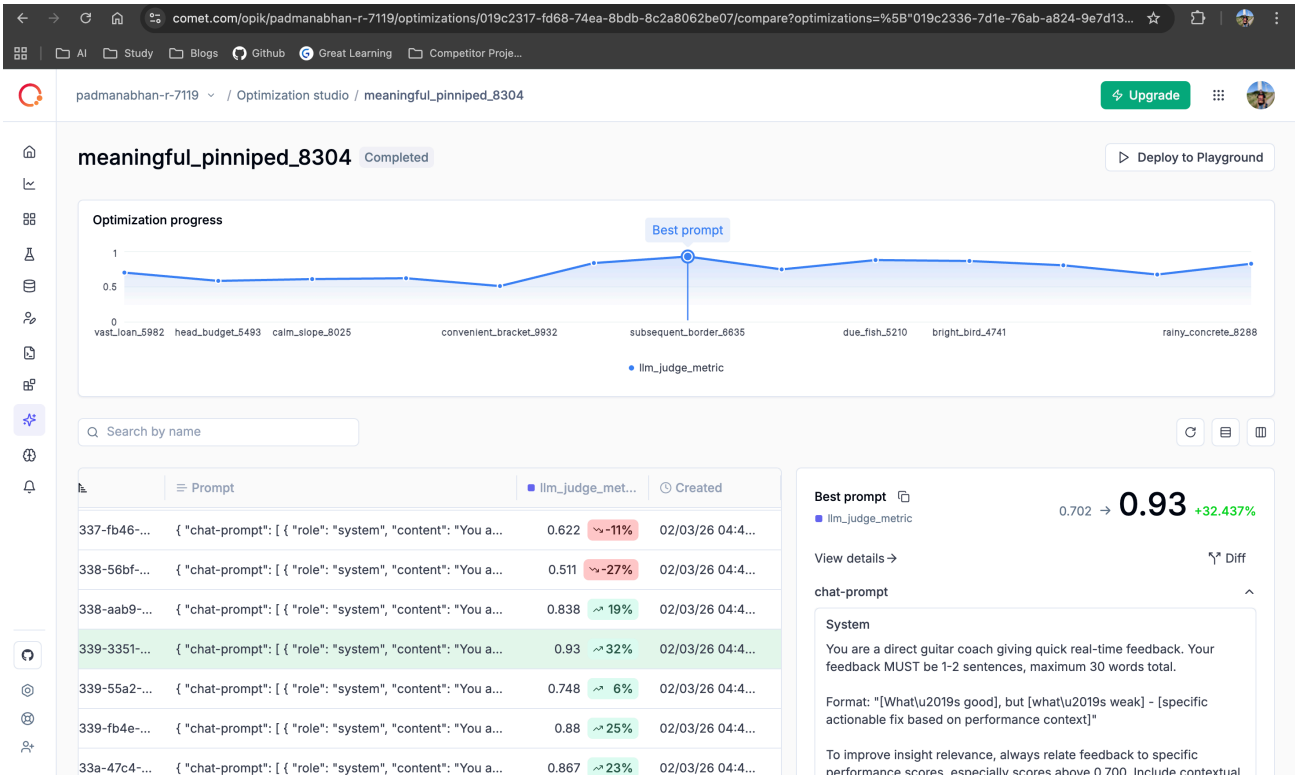
---

## 7. Optimization Studio

Used Optimization Studio to improve the prompt used in the live feedback module.

**Results:**

- 32% increase in `llm_judge_metric` custom metric
- Improved prompt quality for coaching feedback
- Optimized for better real-time guidance




*Optimization Studio results for live feedback prompt*