A Conversational AI Agent for FIB

Hierarchical Design and LLM-as-Judge Evaluation

Ákos Schneider Ignasi Cervero

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Human Language Engineering

The Problem: Fragmented Academic Information

Students face multiple challenges:

- Data scattered across Racó, FIB API, and website
- Simple queries require many clicks
- No system understands implicit context
- Real-time info critical during exams

Example: "When is my next exam?"

→ Check enrolled courses

→ Navigate to exam section

→ Filter by your courses

→ Compare dates manually

Our Solution: Natural Language Interface

A conversational AI agent that:

- Understands natural language queries
- Integrates with the official FIB API
- Handles implicit context automatically
- Supports both public and private data

Now you can just ask:

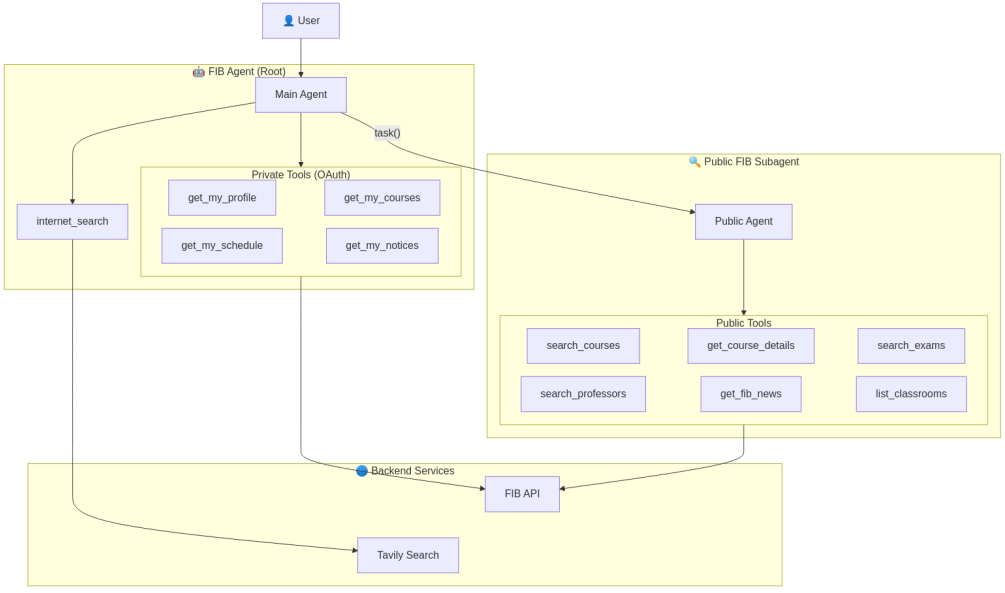
- "When is my next exam?"
- "What do I have tomorrow?"
- "How many credits is IA?"
- "Who teaches EDA?"

Architecture: Hierarchical Agent Design

Two-level hierarchy:

- **Root Agent**
  - User context & authentication
  - Private tools (OAuth)
  - Internet search fallback
  - Task delegation
- **Public Subagent**
  - Specialized for FIB API
  - Focused system prompt
  - Public data queries

Benefits: Focused prompts, security boundary, modularity


User
FIB Agent (Root)
Main Agent
task()
Private Tools (OAuth)
get_my_profile
get_my_courses
get_my_schedule
get_my_notices
internet_search
Public FIB Subagent
Public Agent
Public Tools
search_courses
get_course_details
search_exams
search_professors
get_fib_news
list_classrooms
Backend Services
FIB API
Tavily Search

The Tool Ecosystem

Public Tools (FIB API)

- `search_courses` – Find courses by name/code
- `get_course_details` – Full course info
- `search_exams` – Exam schedules
- `search_professors` – Faculty search
- `get_fib_news` – Announcements
- `list_classrooms` – Room info

Private Tools (OAuth required)

- `get_my_profile` – User info
- `get_my_courses` – Enrolled courses
- `get_my_schedule` – Personal timetable
- `get_my_notices` – Course notices

External

- `internet_search` – Tavily API

All tools are typed Python functions with Pydantic validation

How It Works: The ReAct Loop

Reasoning + Acting paradigm:

1. **Thought** – Reason about the query
2. **Action** – Call appropriate tool
3. **Observation** – Process result
4. **Repeat** until answer ready

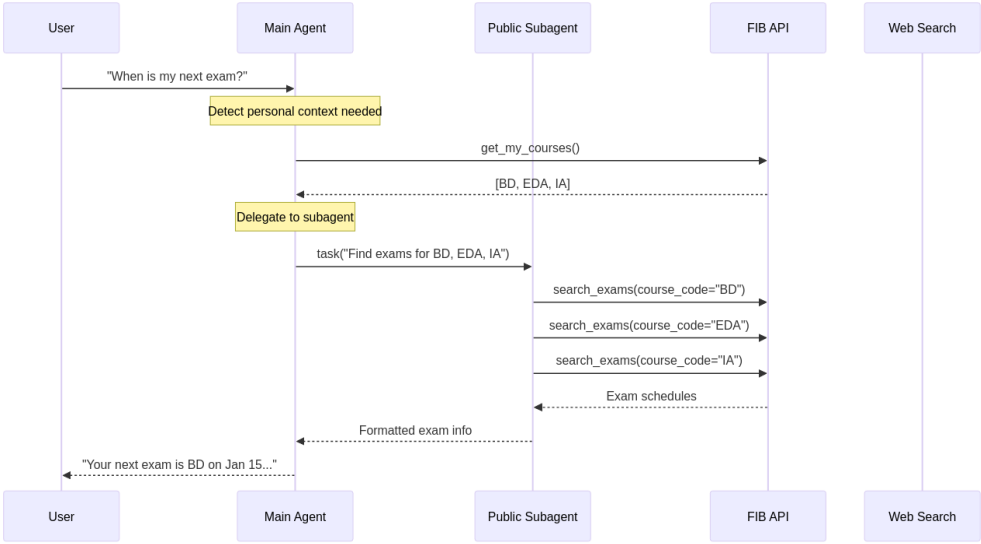
Example thought process:

→ "User wants exam info"

→ "Need their enrolled courses first"

→ Call `get_my_courses()`

→ "Got [BD, EDA, IA], now search exams"


User
Main Agent
Public Subagent
FIB API
Web Search
"When is my next exam?"
Detect personal context needed
get_my_courses()
[BD, EDA, IA]
Delegate to subagent
task("Find exams for BD, EDA, IA")
search_exams(course_code="BD")
search_exams(course_code="EDA")
search_exams(course_code="IA")
Exam schedules
Formatted exam info
"Your next exam is BD on Jan 15..."
User
Main Agent
Public Subagent
FIB API
Web Search

Prompt Engineering Highlights

Structured reasoning framework:

- **Think-Plan-Execute pattern**
  - Analyze query → identify implicit context
  - Plan tool calls → execute systematically
- **Implicit assumption detection**
  - "my exam" → fetch enrolled courses first
  - "tomorrow" → map to weekday number
- **Disambiguation with action**
  - Present top 2–3 matches with context
  - Then ask for clarification (not just list all)
- **Reflect-before-respond checklist**
  - Validate data, check completeness
  - Remove hedging ("it's possible...")

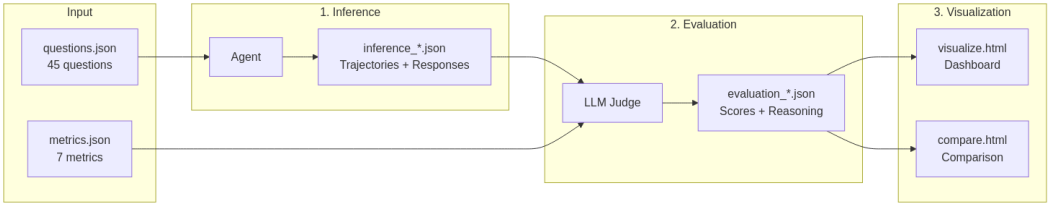
Evaluation: LLM-as-Judge Framework

Why LLM-as-Judge?

- Traditional metrics (BLEU, ROUGE) correlate poorly with quality
- LLM judges understand semantic meaning
- Custom rubrics for each metric

Our setup:

- 45 curated test questions
- 10 categories of queries
- 7 evaluation metrics
- Gemini 2.5 Flash Lite as judge


Input
questions.json
45 questions
metrics.json
7 metrics
1. Inference
Agent
inference_*.json
Trajectories + Responses
2. Evaluation
LLM Judge
evaluation_*.json
Scores + Reasoning
3. Visualization
visualize.html
Dashboard
compare.html
Comparison

Iterative Prompt Refinement with LLM-as-Judge

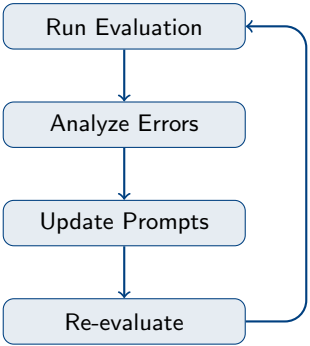
Agent-driven prompt optimization:

1. Run evaluation → identify weak points
2. Coding agent analyzes failure patterns
3. Targeted prompt updates based on errors
4. Re-evaluate → iterate until targets met

Example refinements discovered:

- Added BAD vs GOOD response examples
- Explicit "do NOT use internet search" rules
- Course disambiguation: act first, then clarify
- Weekday mappings to avoid date confusion

This agentic prompt optimization approach is becoming increasingly common in production LLM systems.



Evaluation Dataset & Metrics

Question Categories

Category	Count
courses	13
exams	5
professors	4
personal (OAuth)	4
multi_tool	5
ambiguous	4
news, academic, etc.	10

Complexity: Simple, Multi-step, Contextual, Ambiguous

7 Evaluation Metrics

- **Relevance** – addresses the question?
- **Helpfulness** – actionable & useful?
- **Conciseness** – appropriately brief?
- **Structure** – well-organized?
- **Tone** – professional?
- **Error Handling** – graceful failures?
- **Tool Appropriateness** – right tools?

Results: Model Comparison

Metric	Target	**Gemini 3 Flash**	**Gemini 2.5 Flash**	**Gemini 2.5 Pro**	**Qwen 7B**	**Llama 3B**
Relevance	>0.85	0.98	0.90	0.89	0.74	0.22
Helpfulness	>0.85	0.99	0.89	0.93	0.80	0.32
Conciseness	>0.80	0.87	0.85	0.83	0.69	0.28
Structure	>0.75	0.79	0.77	0.80	0.76	0.29
Tone	>0.80	0.89	0.85	0.85	0.81	0.37
Error Handling	>0.70	0.74	0.51	0.43	0.29	0.08
Tool Approp.	>0.80	0.75	0.77	0.78	0.43	0.09
Average		**0.86**	**0.79**	**0.79**	**0.65**	**0.24**

Key Findings

- Gemini 3 Flash: best overall (0.86 avg), only model to pass error handling
- Gemini 2.5 models: solid performance, Flash & Pro nearly identical
- Open-source: Qwen 7B reasonable (0.65), Llama 3B insufficient (0.24)

Lessons Learned

Challenges & Solutions

- **Ambiguous queries**
  - → Present top matches first
  - → Then ask for clarification
- **Date-relative queries**
  - → Explicit date in context
  - → Weekday mappings in prompt
- **Error handling scored low**
  - → Rubric was overly strict
  - → Agent too verbose on errors
- **Tool tracking across subagents**
  - → Evaluation sees only “task” call
  - → Need trajectory flattening

Contributions & Future Work

Contributions

- Open-source FIB Agent with hierarchical architecture
- 45-question evaluation dataset (10 categories)
- Reusable LLM-as-judge framework
- MCP server for AI interoperability
- Documented prompt engineering patterns

Future Directions

- RAG with syllabi & lecture notes
- Conversation memory (multi-turn)
- User interface (Telegram bot, web chat)
- Multi-university adaptation

Thank You

Key Takeaway

"Modern LLM agents can effectively serve as natural language interfaces to structured APIs when properly configured with domain-specific tools and engineered prompts."

Links

- GitHub: `https://github.com/akossch0/upc-fib-agent`
- FIB API: `https://api.fib.upc.edu/v2/`
- LangGraph: `https://langchain-ai.github.io/langgraph/`