# ClinOrchestra

**Lead/Mentor:** Faculty Lead

**Contributors:** Student Contributors

**Current Funding:**

**Future Funding:**

**IRB #:**

**RMID:**

**SPARCRequest:**

---

## Summary

ClinOrchestra is a **task-agnostic, prompt-agnostic Neuro-Symbolic AI Orchestration Platform** that combines LLM reasoning with deterministic symbolic computation and knowledge retrieval. The platform can support **any task** as long as the output schema is properly defined—from clinical NLP to legal document processing, financial analysis, or any custom domain.

**Key Design Principle:** Users define their output schema, register relevant functions, configure domain-specific RAG sources, and add task-appropriate extras via the Gradio-based web UI using YAML/JSON configuration files. The platform orchestrates the rest.

### Important Research Questions

1. Can neuro-symbolic integration improve task accuracy compared to pure LLM approaches by grounding neural reasoning in symbolic domain knowledge?

2. How does the Neuro-Symbolic Feedback Loop (Functions + RAG + Extras) enhance LLM performance on complex tasks requiring domain expertise?

3. What is the optimal balance between neural (LLM reasoning) and symbolic (deterministic computation) components for different task types?

---

# Architecture

## Core Components

| Component | Purpose |
|---|---|
| **Functions** | Deterministic Python computations (scoring, calculations, lookups) |
| **RAG** | Vector-based document retrieval for domain knowledge grounding |
| **Extras** | Task-specific hints and domain patterns to guide LLM reasoning |

## Execution Modes

| Mode | Description |
|---|---|
| **STRUCTURED** | 4-stage deterministic pipeline: Understanding → Tool Execution → Synthesis → Validation |
| **ADAPTIVE** | ReAct-style iterative agent with self-directed tool selection |

## Data Flow

```
Input Text + Schema → Stage 1: Task Understanding (LLM plans tool usage)
                              ↓
                    Stage 2: Tool Execution (Functions + RAG + Extras in p
                              ↓
                    Stage 3: Synthesis (LLM combines results into structur
                              ↓
                    Stage 4: Validation (Schema check + optional self-crit
                              ↓
                    Final JSON Output
```

---

# Web UI Configuration

All components are configured through the **Gradio web interface** using YAML or JSON files:

## Prompt & Schema Tab

- Define task prompts (main, minimal fallback, RAG refinement)
- Upload JSON schema via YAML/JSON or use interactive field builder

**Schema Example (YAML):**

```
diagnosis:
  type: string
  description: "Primary diagnosis"
  required: true
```

```
severity:
  type: string
  required: true
```

### Functions Tab

- Register Python functions via YAML/JSON upload
- Interactive code editor with parameter builder
- Test execution with JSON arguments

**Functions Example (YAML):**

```
functions:
  - name: calculate_bmi
    description: "Calculate Body Mass Index"
    code: |
      def calculate_bmi(weight_kg, height_m):
          return round(weight_kg / (height_m ** 2), 2)
    parameters:
      weight_kg: {type: number}
      height_m: {type: number}
```

### Extras Tab

- Add domain hints and patterns via YAML/JSON
- Types: `pattern`, `definition`, `guideline`, `example`, `reference`, `criteria`, `tip`

**Extras Example (YAML):**

```
extras:
  - name: "Severity Criteria"
    type: definition
    content: "Mild: score 1-3, Moderate: 4-6, Severe: 7+"
    metadata: {category: clinical}
```

### RAG Tab

- Add document sources: URLs, file paths, or drag-and-drop upload
- Configure: embedding model, chunk size, overlap, K value
- Supported formats: PDF, HTML, TXT, Markdown

---

## Project Experiments

| Experiment | Status | Next Steps |
|---|---|---|
| Neuro-symbolic vs pure LLM comparison | Planned | Design evaluation metrics |
| Function + RAG synergy analysis | Planned | Identify optimal configurations |

| Experiment | Status | Next Steps |
|---|---|---|
| Cross-domain generalization | Planned | Test on non-clinical tasks |

### Next Questions to Answer

1. What performance gains does the neuro-symbolic loop provide over baseline LLM extraction?
2. How does RAG chunk size and K value affect extraction accuracy?
3. Can the platform generalize across domains without architecture changes?

---

## Resources

**Code Repository:** GitHub (private/institutional)

**Key Dependencies:** - `anthropic` / `openai` / `ollama` - LLM backends - `sentence-transformers` - Embedding generation - `faiss-cpu` / `faiss-gpu` - Vector similarity search - `gradio` - Web UI framework - `pydantic` - Schema validation

**Datasets Used:** To be specified per experiment

---


## Acknowledgements

**Lead/Mentor:** Faculty Lead

**Contributors:** Student Contributors

**Past Contributors:**

**Current Funding:**

**Future Funding:**

**Institution:** Medical University of South Carolina, Biomedical Informatics Center