AI Agent Engr Task 2 :Autonomous AI Tutor Orchestrator

Problem Title

Intelligent Multi-Tool Orchestration for Autonomous AI Tutoring Systems

Problem Statement

Build an **intelligent middleware orchestrator** that can autonomously connect a conversational AI tutor to multiple educational tools by extracting required parameters from chat context and managing complex tool interactions without manual configuration.

Your system should act as the “brain” that sits between a student’s conversation with an AI tutor and the actual educational tools (quiz generators, note makers, concept explainers, etc.). The core challenge is to create a system that can:

1. **Understand conversational context** and determine what educational tools are needed
2. **Intelligently extract parameters** required by various tools from natural conversation
3. **Validate and format requests** to ensure proper tool execution
4. **Handle diverse tool schemas** across multiple educational functionalities
5. **Maintain conversation state** and student personalization context

You are NOT building the tutor interface or the educational tools themselves - your focus is purely on the intelligent orchestration layer that makes autonomous tool selection and execution possible.

Core Challenge

The primary technical challenge lies in creating a system that can handle **80+ different educational tools**, each with unique input/output schemas, and automatically fill their required parameters based on conversational context rather than explicit user input.

Example Scenario

```
Student: "I'm struggling with calculus derivatives and need some practice pro
blems"

Your System Should:
1. Determine that a "Quiz Generator" tool is appropriate
2. Extract parameters:
   - topic: "derivatives"
   - subject: "calculus"
   - difficulty: "beginner" (inferred from "struggling")
   - question_type: "practice" (inferred from "practice problems")
   - num_questions: [use default/infer from context]
3. Validate parameter schema and make API call
4. Handle response and format for tutor presentation
```

Technical Requirements

Technology Stack

- **Backend**: Python with FastAPI
- **Agent Framework**: LangGraph and LangChain as primary frameworks
- **Database**: PostgreSQL (if persistent storage needed)
- **Additional**: You may integrate other complementary frameworks

Core Functionality Requirements

1. Context Analysis Engine

- Parse conversation history and current student message
- Identify educational intent and tool requirements
- Extract relevant entities (topics, subjects, difficulty levels, etc.)

2. Parameter Extraction System

- Map conversational elements to specific tool parameters
- Handle missing parameters through intelligent inference or targeted questions
- Validate extracted parameters against tool schemas

3. Tool Orchestration Layer

- Manage API calls to multiple educational tools
- Handle different authentication methods and rate limiting
- Process and normalize responses from various tools

4. State Management

- Maintain conversation context across interactions
- Track student preferences and learning state
- Store and retrieve relevant session information

5. Schema Validation

- Ensure API requests meet tool specifications
- Handle validation errors gracefully
- Provide meaningful error messages for debugging

Implementation Approaches

You may choose from several architectural approaches (or combine them):

Approach 1: Direct API Integration

Create a unified API layer that directly connects to educational tool endpoints through standardized request/response handling.

Approach 2: MCP Server Architecture

Implement Model Context Protocol (MCP) servers that expose educational tools through standardized interfaces for LLM agent consumption.

Approach 3: Middleware API Pattern

Build a middleware service that acts as an intelligent proxy between the conversational agent and educational tools.

Approach 4: Hybrid Agent System

Combine multiple specialized agents (parameter extraction, tool selection, validation) coordinated through LangGraph workflows

Approach 5: Design Your Own Solution

Feel free to propose and implement your own architectural approach that addresses the core challenges. Innovation is encouraged, and you may combine elements from different approaches or introduce entirely new patterns that effectively solve the orchestration problem.

Utilizing Educational Context Data

Your system must incorporate educational context data that influences how tool parameters are extracted, requests are formed, and responses are customized. This includes:

Teaching Styles

Participants will be provided with descriptions of teaching styles such as **Direct, Socratic, Visual, and Flipped Classroom**.
Your orchestrator should show how it would adapt the content or parameter choices based on a chosen or inferred teaching style. For example:

- For a **Direct** style, provide concise and factual explanations with minimal elaboration.
- For a **Visual** style, parameters may request more examples or analogies to enhance understanding.

Emotional States

Emotional states such as **Focused, Anxious, Confused, and Tired** will be provided as keywords. Your system should demonstrate the ability to adapt:

- If the student is **Confused**, send simplified versions of content or break concepts into smaller components when calling tools.
- If **Focused**, the system might increase difficulty or present more challenging questions.

Mastery Levels

Based on a 10-level mastery framework from foundation to full mastery, adapt parameter filling accordingly. For example, a student with a low mastery level might receive beginner-level questions or requests for fundamental explanations, while advanced students receive nuanced or accelerated content.

Resources Provided

API Documentation Package

1. Note Maker Tool

Input Schema

```
{
  "type": "object",
  "required": ["user_info", "chat_history", "topic", "subject", "note_taking_styl
e"],
  "properties": {
    "user_info": {
      "type": "object",
      "required": ["user_id", "name", "grade_level", "learning_style_summary",
  "emotional_state_summary", "mastery_level_summary"],
      "properties": {
        "user_id": {
          "type": "string",
          "description": "Unique identifier for the student",
          "example": "student123"
        },
        "name": {
          "type": "string",
          "description": "Student's full name",
          "example": "Harry"
        },
        "grade_level": {
          "type": "string",
          "description": "Student's current grade level",
          "example": "10"
        },
        "learning_style_summary": {
          "type": "string",
          "description": "Summary of student's preferred learning style",
          "example": "Prefers outlines and structured notes"
        },
        "emotional_state_summary": {
          "type": "string",
          "description": "Current emotional state of the student",
          "example": "Relaxed and attentive"
        },
        "mastery_level_summary": {
          "type": "string",
          "description": "Current mastery level description",
          "example": "Level 7: Proficient"
        }
      }
    },
    "chat_history": {
      "type": "array",
      "description": "Recent conversation history to provide context",
      "items": {
        "type": "object",
        "properties": {
          "role": {
            "type": "string",
            "enum": ["user", "assistant"],
            "description": "Role of the message sender"
          },
          "content": {
            "type": "string",
            "description": "Content of the message"
          }
        }
      }
    },
    "topic": {
      "type": "string",
      "description": "The main topic for note generation",
      "example": "Water Cycle"
    },
    "subject": {
      "type": "string",
      "description": "Academic subject area",
      "example": "Environmental Science"
    },
    "note_taking_style": {
      "type": "string",
      "enum": ["outline", "bullet_points", "narrative", "structured"],
      "description": "Preferred format for the notes"
    },
    "include_examples": {
      "type": "boolean",
      "description": "Whether to include examples in the notes",
      "default": true
    },
    "include_analogies": {
      "type": "boolean",
      "description": "Whether to include analogies in the notes",
      "default": false
    }
  }
}
```

Parameter Descriptions

Required Parameters

- **user_info**: Student profile information containing:
  - **user_id**: Unique student identifier (string, required)
  - **name**: Student's name (string, required)
  - **grade_level**: Current grade level (string, e.g., "10", "Grade 10")
  - **learning_style_summary**: Brief description of learning preferences
  - **emotional_state_summary**: Current emotional/motivational state
  - **mastery_level_summary**: Current skill level description
- **chat_history**: Array of recent conversation messages providing context
- **topic**: Main subject for note generation (string)
- **subject**: Academic discipline (string)
- **note_taking_style**: Output format - must be one of: "outline", "bullet_points", "narrative", "structured"

Optional Parameters

- **include_examples**: Boolean flag to include practical examples (default: true)

- **include_analogies**: Boolean flag to include explanatory analogies (default: false)

Response Format

Success Response (200)

```
{
  "type": "object",
  "properties": {
    "topic": {
      "type": "string",
      "description": "The topic of the generated notes"
    },
    "title": {
      "type": "string",
      "description": "Generated title for the notes"
    },
    "summary": {
      "type": "string",
      "description": "Brief overview of the note content"
    },
    "note_sections": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "title": {
            "type": "string"
          },
          "content": {
            "type": "string"
          },
          "key_points": {
            "type": "array",
            "items": {
              "type": "string"
            }
          },
          "examples": {
            "type": "array",
            "items": {
              "type": "string"
            }
          },
          "analogies": {
            "type": "array",
            "items": {
              "type": "string"
            }
          }
        }
      }
    },
    "key_concepts": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "connections_to_prior_learning": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "visual_elements": {
      "type": "array",
      "items": {
        "type": "object"
      }
    },
    "practice_suggestions": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "source_references": {
      "type": "array",
      "items": {
        "type": "string"
      }
    },
    "note_taking_style": {
      "type": "string"
    }
  }
}
```

2. Flashcard Generator Tool

Input Schema

```
{
  "type": "object",
  "required": ["user_info", "topic", "count", "difficulty", "subject"],
  "properties": {
    "user_info": {
      "type": "object",
      "required": ["user_id", "name", "grade_level", "learning_style_summary",
"emotional_state_summary", "mastery_level_summary"],
      "properties": {
        "user_id": {
          "type": "string",
          "description": "Unique identifier for the student",
          "example": "student123"
        },
        "name": {
          "type": "string",
          "description": "Student's full name",
          "example": "Charlie"
        },
        "grade_level": {
          "type": "string",
          "description": "Student's current grade level",
          "example": "8"
        },
        "learning_style_summary": {
          "type": "string",
          "description": "Summary of student's preferred learning style",
          "example": "Kinesthetic learner, learns best through practice and repetit
ion"
        },
        "emotional_state_summary": {
          "type": "string",
          "description": "Current emotional state of the student",
          "example": "Focused and motivated to improve"
        },
        "mastery_level_summary": {
          "type": "string",
          "description": "Current mastery level description",
          "example": "Level 6: Good understanding, ready for application"
        }
      }
    },
    "topic": {
      "type": "string",
      "description": "The topic for flashcard generation",
      "example": "Photosynthesis"
    },
    "count": {
      "type": "integer",
      "description": "Number of flashcards to generate",
      "minimum": 1,
      "maximum": 20,
      "example": 5
    },
    "difficulty": {
      "type": "string",
      "enum": ["easy", "medium", "hard"],
      "description": "Difficulty level of the flashcards",
      "example": "medium"
    },
    "include_examples": {
      "type": "boolean",
      "description": "Whether to include examples in flashcards",
      "default": true
    },
    "subject": {
      "type": "string",
      "description": "Academic subject area",
      "example": "Biology"
    }
  }
}
```

Parameter Descriptions

Required Parameters

- **user_info**: Student profile information (same structure as Note Maker)
- **topic**: Topic for flashcard generation (string)
- **count**: Number of flashcards to create (integer, 1-20 range)
- **difficulty**: Difficulty level - must be one of: "easy", "medium", "hard"
- **subject**: Academic discipline (string)

Optional Parameters

- **include_examples**: Boolean flag to include examples (default: true)

Response Format

Success Response (200)

```
{
  "type": "object",
  "properties": {
    "flashcards": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "title": {
            "type": "string",
            "description": "Title/category of the flashcard"
          },
          "question": {
            "type": "string",
            "description": "Question side of the flashcard"
          },
          "answer": {
            "type": "string",
            "description": "Answer side of the flashcard"
          },
          "example": {
            "type": "string",
            "description": "Optional example or explanation"
          }
        }
      }
    },
    "topic": {
      "type": "string",
      "description": "Topic of the flashcards"
    },
    "adaptation_details": {
      "type": "string",
      "description": "Information about how flashcards were adapted for the stu
dent"
    },
    "difficulty": {
      "type": "string",
      "description": "Difficulty level used"
    }
  }
}
```

3. Concept Explainer Tool

Input Schema

```
{
  "type": "object",
  "required": ["user_info", "chat_history", "concept_to_explain", "current_topi
c", "desired_depth"],
  "properties": {
    "user_info": {
      "type": "object",
      "required": ["user_id", "name", "grade_level", "learning_style_summary",
"emotional_state_summary", "mastery_level_summary"],
      "properties": {
        "user_id": {
          "type": "string",
          "description": "Unique identifier for the student",
          "example": "student123"
        },
        "name": {
          "type": "string",
          "description": "Student's full name",
          "example": "Bob"
        },
        "grade_level": {
          "type": "string",
          "description": "Student's current grade level",
          "example": "7"
        },
        "learning_style_summary": {
          "type": "string",
          "description": "Summary of student's preferred learning style",
          "example": "Auditory learner, prefers simple terms and step-by-step exp
lanations"
        },
        "emotional_state_summary": {
          "type": "string",
          "description": "Current emotional state of the student",
          "example": "Curious and engaged in learning"
        },
        "mastery_level_summary": {
          "type": "string",
          "description": "Current mastery level description",
          "example": "Level 4: Building foundational knowledge"
        }
      }
    },
    "chat_history": {
      "type": "array",
      "description": "Recent conversation history for context",
      "items": {
        "type": "object",
        "properties": {
          "role": {
            "type": "string",
            "enum": ["user", "assistant"]
          },
          "content": {
            "type": "string"
          }
        }
      }
    },
    "concept_to_explain": {
      "type": "string",
      "description": "The specific concept to explain",
      "example": "photosynthesis"
    },
    "current_topic": {
      "type": "string",
      "description": "Broader topic context",
      "example": "biology"
    },
    "desired_depth": {
      "type": "string",
      "enum": ["basic", "intermediate", "advanced", "comprehensive"],
      "description": "Level of detail for the explanation",
      "example": "medium"
    }
   }
}
```

Parameter Descriptions

Required Parameters

- **user_info**: Student profile information (same structure as other tools)
- **chat_history**: Array of recent conversation messages
- **concept_to_explain**: Specific concept to explain (string)
- **current_topic**: Broader subject context (string)

- **desired_depth**: Detail level - must be one of: "basic", "intermediate", "advanced", "comprehensive"

Response Format

Success Response (200)

```
{
  "type": "object",
  "properties": {
    "explanation": {
      "type": "string",
      "description": "Main explanation text"
    },
    "examples": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "Practical examples illustrating the concept"
    },
    "related_concepts": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "Related concepts for further learning"
    },
    "visual_aids": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "Suggestions for visual representations"
    },
    "practice_questions": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "Questions for practice and assessment"
    },
    "source_references": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "References for additional study"
    }
  }
}
```

Common Data Types and Validation Rules

User Info Object

All three tools require a consistent `user_info` object with these validation rules:

- **user_id**: Non-empty string, unique identifier
- **name**: Non-empty string, student's name
- **grade_level**: String representing grade (e.g., "7", "Grade 10", "11")
- **learning_style_summary**: Non-empty string describing learning preferences
- **emotional_state_summary**: Non-empty string describing current emotional state
- **mastery_level_summary**: Non-empty string with level description (e.g., "Level 4: Building foundational knowledge")

Chat History Array

- Array of message objects with `role` and `content` properties

- `role` must be either "user" or "assistant"
- `content` should be non-empty strings

Validation Constraints

- String lengths: Most text fields should be reasonable (under 1000 characters for summaries)
- Array sizes: Chat history typically 1-10 messages, flashcards 1-20 items
- Boolean values: Standard true/false for flags
- Enum values: Strict validation on predefined options

Error Handling

All endpoints follow consistent error response patterns:

- **400 Bad Request**: Invalid input parameters
- **401 Unauthorized**: Authentication issues
- **404 Not Found**: Resource not found
- **429 Too Many Requests**: Rate limiting
- **500 Internal Server Error**: Server-side issues

Error responses include:

- `error`: Human-readable error message
- `error_code`: Machine-readable error code
- `details`: Optional additional error information

Educational Context Data

Teaching Styles

- **Direct**: Clear, concise, step-by-step instruction
- **Socratic**: Question-based guided discovery learning
- **Visual**: Descriptive imagery and analogical explanations

- **Flipped Classroom**: Application-focused with assumed prior knowledge

Emotional States

- **Focused/Motivated**: High engagement, ready for challenges
- **Anxious**: Needs reassurance and simplified approach
- **Confused**: Requires clarification and step-by-step breakdown
- **Tired**: Minimal cognitive load, gentle interaction

Mastery Levels

- **Levels 1-3**: Foundation building with maximum scaffolding
- **Levels 4-6**: Developing competence with guided practice
- **Levels 7-9**: Advanced application and nuanced understanding
- **Level 10**: Full mastery enabling innovation and teaching others

Sample Data

- **Mock student profiles** with various learning states and preferences

Success Criteria

Your solution will be evaluated based on:

1. Parameter Extraction Accuracy (40%)

- Correctly identify required tool parameters from conversational context
- Handle ambiguous or incomplete information appropriately
- Demonstrate intelligent inference for missing parameters

2. Tool Integration Completeness (25%)

- Successfully integrate with multiple provided educational tools
- Handle different tool schemas and response formats
- Implement proper error handling and validation

3. System Architecture (20%)

- Design scalable architecture suitable for 80+ tool integration
- Implement clean separation of concerns
- Demonstrate understanding of chosen architectural approach

4. User Experience (10%)

- Natural conversation flow without disrupting student interaction
- Graceful handling of edge cases and errors
- Clear demonstration of autonomous operation

5. Technical Implementation (5%)

- Code quality, documentation, and best practices
- Proper use of required frameworks (LangGraph, LangChain)
- Performance considerations and optimization

Submission Requirements

Deliverable Format

Submit a **Google Document** containing:

1. **Solution Overview** (1-2 pages)
    - High-level architecture description
    - Chosen implementation approach and rationale
    - Key technical decisions and trade-offs
2. **Implementation Documentation** (2-3 pages)
    - Detailed system design with diagrams
    - Parameter extraction methodology
    - Tool integration patterns
    - State management approach

3. **Demo Documentation** (Video)
    - Step-by-step demo scenario walkthrough
    - Expected inputs and outputs
    - Error handling demonstrations
4. **Technical Appendix**
    - Links to code repository (GitHub/GitLab)
    - API documentation for your endpoints
    - Setup and deployment instructions
    - Performance metrics and testing results

Code Repository Structure

Your repository should include:
- **Source code** with clear module organization
- **Configuration files** for environment setup
- **API documentation** (OpenAPI/Swagger specs)
- **Test suite** with unit and integration tests
- **README** with setup and usage instructions

Demonstration Requirements

Prepare a 5**-minute demo** showcasing:
1. Natural conversation leading to tool parameter extraction
2. Successful execution of multiple educational tools
3. Handling of ambiguous or incomplete user input
4. System response to validation errors or edge cases

Personalization Context

Your system should handle basic personalization parameters:

Student State Variables

- **Current mastery level** (1-10 scale based on provided framework)
- **Emotional state** (focused, anxious, confused, tired)

- **Preferred teaching style** (direct, socratic, visual, flipped classroom)
- **Learning goals** and subject preferences

Context Adaptation

You may use **static values** for personalization parameters during development, but your system should demonstrate how these parameters would influence:
- Tool selection decisions
- Parameter value inference
- Response formatting and tone

Important Notes

What You DON'T Need to Build

- ❌ The conversational AI tutor interface
- ❌ The educational tools themselves (quiz generators, note makers, etc.)
- ❌ Advanced personalization engines or learning analytics
- ❌ Tool selection logic (this is provided)
- ❌ Complete database integration (focus on core orchestration)

What You DO Need to Focus On

- ✅ Intelligent parameter extraction from conversations
- ✅ Robust tool integration and schema handling
- ✅ Request validation and error management
- ✅ Scalable architecture for multiple tools
- ✅ Clean demonstration of autonomous operation

Questions or Clarifications

For technical questions during development, please refer to:
- **API Documentation** provided in the resource package
- **Sample implementations** and use cases in the starter materials
- **Technical support channels** that will be available during the hackathon

Ready to build the future of autonomous AI education? Let's see your intelligent orchestration solution in action! 🚀

Appendix: Available Teaching Styles and States

Teaching Styles

- **Direct**: Clear, concise, step-by-step instruction
- **Socratic**: Question-based guided discovery learning
- **Visual**: Descriptive imagery and analogical explanations
- **Flipped Classroom**: Application-focused with assumed prior knowledge

Emotional State

- **Focused/Motivated**: High engagement, ready for challenges
- **Anxious**: Needs reassurance and simplified approach
- **Confused**: Requires clarification and step-by-step breakdown
- **Tired**: Minimal cognitive load, gentle interaction

Mastery Levels

- **Levels 1-3**: Foundation building with maximum scaffolding
- **Levels 4-6**: Developing competence with guided practice
- **Levels 7-9**: Advanced application and nuanced understanding
- **Level 10**: Full mastery enabling innovation and teaching others