Multi-Agent System for Energy Modeling and Analysis

System Overview

This multi-agent framework implements a coordinated system of specialized AI agents that work together to process user requests, build energy models, analyze results, and generate reports. The system uses a central coordinator (Nova) that delegates tasks to specialized agents (Emil, Ivan, and Lola) based on the intent of user prompts. The framework supports both synchronous and asynchronous operation, allows for multi-prompt processing, and maintains conversation history for context-aware responses.

System Architecture

The architecture follows a coordinated multi-agent design pattern with the following components:

1. **Agent Layer**: Specialized agents with distinct capabilities
    - **Nova**: Central coordinator for intent detection and task delegation
    - **Emil**: Energy modeling specialist
    - **Ivan**: Code generation and image creation specialist
    - **Lola**: Report writing and copywriting specialist
2. **Core Components**:
    - **Knowledge Base**: Shared data repository for all agents
    - **Task Manager**: Task creation, tracking, and execution system
    - **Function Registry**: Registration and mapping of available functions
3. **Utility Layer**:
    - **LLM Integration**: OpenAI and other LLM service connectors
    - **Function Mapping**: CSV-based function mapping system
    - **Parameter Collection**: User interaction for gathering required parameters
    - **Specialized Utilities**: Math processing, general knowledge, etc.
4. **Main Application**:
    - Entry point with both synchronous and asynchronous operation modes
    - Session management and history tracking
    - User interface for prompt input and result display

Component Details

Agent Layer

Agent	File	Primary Responsibilities	Key Features
Base Agent	`src/agents/base_agent.py`	Define common interface for all agents	Task handling, parameter collection, KB interaction

Nova	src/agents/nova.py	Coordinate overall system, detect intents, delegate tasks	Intent detection, task creation, task routing
Emil	src/agents/emil.py	Energy modeling, model analysis	Parameter verification, model creation, result analysis
Ivan	src/agents/ivan.py	Code generation, image creation	Python scripting, DALL-E integration, ASCII fallbacks
Lola	src/agents/lola.py	Report writing, copywriting	Multiple report styles, analysis integration, content formatting

Core Components

Component	File	Primary Responsibilities	Key Features
Knowledge Base	src/core/knowledge_base.py	Centralized data storage	Persistence, categorization, session tracking
Task Manager	src/core/task_manager.py	Task creation and execution	Hierarchical tasks, status tracking, result storage
Function Registry	src/core/functions_registery.py	Define core system functions	Energy modeling, analysis, reporting functions
Main Application	src/main.py	System entry point and coordination	Multiple operation modes, session management

Utility Layer

Utility	File	Primary Responsibilities	Key Featur
CSV Function Mapper	src/utils/csv_function_mapper.py	Map functions from CSV to implementations	Configuratio driven approach, flexible registration
Math Processor	src/utils/do_maths.py	Handle mathematical calculations	Local patter matching, LL fallback
Function Logger	src/utils/function_logger.py	Log function calls	Simple decorator implementati

General Knowledge	src/utils/general_knowledge.py	Answer general questions, handle history	History detection, context-awar responses
OpenAI Calls	src/utils/open_ai_calls.py	Direct OpenAI API interaction	Multiple mod support, err handling
OpenAI Utilities	src/utils/open_ai_utils.py	Enhanced OpenAI integration	Async suppor categorizati standardized interfaces
Parameter Collection	src/utils/simplified_parameter_collection.py	Collect parameters from users	Clear descriptions consistent interface

System Workflow

1. **Initialization**:
    - Load the knowledge base, potentially with previous session data
    - Register available functions with the function registry
    - Initialize all agent instances with their function maps
    - Create a new session for tracking interactions
2. **Prompt Processing**:
    - User enters one or more prompts
    - Nova processes each prompt to identify intents
    - Nova categorizes each intent to determine the appropriate agent
    - Nova creates tasks for each intent with necessary parameters
3. **Task Execution**:
    - Tasks are routed to the appropriate specialized agents
    - Complex workflows (like energy modeling + analysis + reporting) are ordered correctly
    - Agents execute their assigned tasks, collecting additional parameters if needed
    - Results are stored in the knowledge base and returned to the task manager
4. **Result Presentation**:
    - Results from all tasks are collected and formatted
    - Combined output is presented to the user
    - Session data is stored for future context
5. **Session Management**:
    - Interaction history is maintained for context-aware responses
    - Session data is categorized for efficient retrieval
    - Old sessions are archived to manage storage

Key Technologies

1. **Language Models**:
   - OpenAI API for intent detection, categorization, and general knowledge
   - Support for multiple models (o3-mini, GPT models, etc.)
   - Local model options for development and testing
2. **Asynchronous Processing**:
   - AsyncIO for concurrent task processing
   - Thread pools for running synchronous functions in async contexts
   - Lock mechanisms for thread safety in knowledge base operations
3. **Persistent Storage**:
   - JSON-based storage for knowledge base persistence
   - Automatic backups on hourly intervals
   - Session archiving for long-term storage management
4. **Function Mapping**:
   - CSV-based function definitions for flexibility
   - Dynamic mapping between function names and implementations
   - Support for enhanced function maps with additional metadata
5. **Error Handling**:
   - Comprehensive error detection and reporting
   - Fallback mechanisms for API failures
   - Detailed logging for debugging and monitoring

Interaction Flow

```
User
 │
 ▼
┌───────────┐     ┌───────────┐     ┌───────────┐
│           │     │           │     │           │
│   Nova    │◄───►│ Knowledge │◄───►│   Task    │
│           │     │   Base    │     │  Manager  │
│           │     │           │     │           │
└───────────┘     └───────────┘     └───────────┘
    │                                    │
    │                                    │
    ▼                                    ▼
┌───────────┐     ┌───────────┐     ┌───────────┐
│           │     │           │     │           │
│   Emil    │     │   Ivan    │     │   Lola    │
│           │     │           │     │           │
└───────────┘     └───────────┘     └───────────┘
    │                 │                 │
    └─────────────────┼─────────────────┘
                      │
```

```
                        ▼
                      User
```

Extension Points

The system is designed for extensibility in several key areas:

1. **New Agents**: Additional specialized agents can be added by extending the BaseAgent class
2. **New Functions**: New capabilities can be added through the function registry and CSV mappings
3. **New LLM Providers**: The OpenAI utilities are designed to support multiple LLM providers
4. **Enhanced Persistence**: The knowledge base can be extended with different storage backends
5. **Additional Parameters**: The parameter collection system can be extended with new parameter types

Conclusion

This multi-agent framework provides a flexible, extensible system for energy modeling and analysis through a coordinated set of specialized agents. By separating concerns into distinct agents and using a central coordinator, the system can handle complex workflows while maintaining a clean architecture. The asynchronous capabilities enable efficient processing of multiple requests, while the persistent knowledge base ensures continuity across sessions.