Project - High Level Design

On

Multi-Agent Education System

Course Name: Agentic AI

Institution Name: Medicaps University – Datagami Skill Based Course

Student Name(s) & Enrolment Number(s):

Sr no	Student Name	Enrolment Number
1	Priywrat Singh Dodiya	EN22CS301771
2	Rohan Patil	EN22CS301818
3	Priyansh Maheshwari	EN22CS301762
4	Rohit Chahar	EN22CS301821
5	Parth Rai	EN22CS301693

Group Name: 01D7

Project Number: AAI-24

Industry Mentor Name:

University Mentor Name: Prof. Ajeet Singh Rajput

Academic Year: 2026

1. Introduction

Artificial Intelligence has significantly evolved from rule-based automation systems to intelligent autonomous systems capable of reasoning, planning, and decision-making. In the education sector, AI technologies are being used to build intelligent tutoring systems, automated grading platforms, adaptive learning environments, and content generation tools. However, most traditional AI-powered educational systems rely on a single large language model (LLM) to perform all tasks including understanding, researching, organizing, and formatting content. While effective for short responses, this approach becomes limited when handling complex academic topics that require structured reasoning and controlled workflow execution.

The system consists of three primary agents:

• **Researcher Agent** – Responsible for analyzing the user query, breaking it into logical subtopics, retrieving relevant information, and organizing structured research data.

• **Writer Agent** – Responsible for transforming structured research data into formatted, coherent, and academically organized content.

• **Orchestrator Agent** – Responsible for managing workflow, coordinating agent communication, delegating tasks, and maintaining execution state.

This system demonstrates:

• Multi-agent collaboration

• Task specialization

• Structured agent hand-off

• Context-aware reasoning

• Tool-augmented execution

• Modular and scalable design

The primary objective of this system is to generate structured, high-quality educational content using collaborative intelligent agents that operate under controlled workflow execution.

2. Scope of the Document

This document describes the High-Level Design (HLD) of the Multi-Agent Education System. It provides a complete architectural blueprint explaining how system components interact and how multi-agent collaboration is structured.

In Scope:

• System architecture and layered design

• Agent roles and responsibilities

• Process flow and information flow

• Data model and lifecycle management

• API catalogue and communication structure

• State and session management

• Caching strategy

• Security and performance considerations

• Scalability and maintainability

Out of Scope:

• Detailed source code implementation

• Frontend UI design

• Deployment configuration scripts

• Production infrastructure setup

• Monitoring and logging implementation

The focus remains strictly on architectural clarity and conceptual system design.

3. Intended Audience

This document is intended for:

• Final year B.Tech CSE students

• Internship project evaluators

• Faculty mentors

• System reviewers

• AI researchers

Readers are expected to have basic knowledge of:

• Artificial Intelligence fundamentals

• Python programming

• REST API-based systems

• Large Language Models (LLMs)
• Software architecture principles

4. System Overview

The Multi-Agent Education System is a collaborative AI framework designed to process educational queries and produce structured academic content.

For example, when a user submits:

"Explain Operating System Scheduling Algorithms with examples."

The system performs the following stages:

1. **Query Understanding** – Interpret user intent and identify expected output structure.
2. **Topic Decomposition** – Break the topic into logical subtopics.
3. **Resource Retrieval** – Collect relevant information using reasoning and tools.
4. **Content Synthesis** – Organize information into structured academic format.
5. **Output Formatting** – Present content with headings, explanations, and examples.

The system ensures that content generation is controlled, structured, and logically sequenced rather than generated randomly in a single reasoning step.

5. System Architecture

The system follows a **Layered Multi-Agent Modular Architecture**. This design ensures separation of concerns, scalability, and independent development of system components.

Architectural Layers:

• Presentation Layer
• API Layer
• Orchestration Layer
• Agent Layer
• Intelligence Layer
• Tools Layer
• Data & Session Layer

5.1 High-Level Architecture Diagram (Logical View)

User

↓

Frontend Interface

↓

Backend API

↓

Orchestrator Agent

↓

Researcher Agent → Tools

↓

Writer Agent → LLM API

↓

Final Output

This logical view highlights structured delegation and coordination between system components.

6. Application Design

The system is divided into clearly defined layers to maintain modularity.

6.1 Presentation Layer

• Accepts user queries

• Displays structured educational output

• Provides user interaction interface

6.2 API Layer

• Handles HTTP requests

• Validates input

• Routes queries to Orchestrator

• Formats responses

6.3 Orchestration Layer

• Controls workflow sequencing

• Delegates tasks between agents

• Maintains session state

• Handles error recovery

6.4 Agent Layer

• Researcher Agent

• Writer Agent

Each agent operates autonomously within assigned responsibilities.

6.5 Intelligence Layer

• LLM integration

• Prompt management

• Structured reasoning logic

6.6 Tools Layer

• Web search integration

• Document parsing utilities

• Knowledge database

• Optional vector database

7. Process Flow

Step-by-step workflow:

1. User submits educational query.
2. Backend API validates request.
3. Orchestrator initializes session context.
4. Researcher Agent performs topic breakdown.
5. Researcher retrieves structured knowledge.
6. Research output returned to Orchestrator.
7. Writer Agent generates structured content.
8. Final output returned to user.

This structured sequence ensures proper reasoning order.

8. Sequence Diagram

User → Frontend → Backend → Orchestrator → Researcher → Tools → Orchestrator → Writer → LLM → Orchestrator → Backend → User

This sequence ensures controlled communication and prevents uncontrolled execution.

9. Components Design

9.1 Orchestrator Agent

Responsibilities:

• Workflow control

• Agent invocation

• State management

• Error handling

• Response aggregation

The Orchestrator ensures deterministic execution.

9.2 Researcher Agent

Responsibilities:

• Query analysis

• Topic decomposition

• Sub-topic identification

• Knowledge retrieval

• Data structuring

The Researcher ensures completeness and logical organization.

9.3 Writer Agent

Responsibilities:

• Content generation

• Heading creation

• Paragraph structuring

• Example inclusion

• Formatting

The Writer ensures clarity and academic presentation.

9.4 Tools Module

Includes:

• Web search API

• Document parser

• Knowledge retrieval system

• Validation mechanisms

Tools reduce hallucination and improve reliability.

10. Data Design

10.1 Data Model

Entity	Description
Query	User request
ResearchData	Structured retrieved knowledge
DraftContent	Intermediate structured output
FinalOutput	Completed formatted content
AgentState	Workflow tracking

10.2 Data Flow

User Query

→ ResearchData

→ DraftContent

→ FinalOutput

10.3 Data Retention

• Temporary session storage

• Optional output storage

• Logging for debugging

11. API Catalogue

API	Method	Description
/submit	POST	Submit query
/research	POST	Trigger research
/write	POST	Trigger writing
/status	GET	Session status

API	Method	Description
/result	GET	Final output

12. State and Session Management

Maintains:

• Current query context

• Research completion flag

• Writing completion flag

• Error state

Enables:

• Multi-step reasoning

• Context preservation

• Recovery from failures

13. Caching Strategy

• Query caching

• Research caching

• LLM response caching

Benefits:

• Faster response

• Reduced cost

• Improved scalability

14. Non-Functional Requirements

14.1 Security

• Input validation

• Secure API keys

• Controlled tool access

• Secure communication

14.2 Performance

• Efficient prompt design

• Reduced redundant calls

• Predictable latency

14.3 Scalability

• Modular architecture

• Easy addition of agents

• Horizontal scaling capability

14.4 Maintainability

• Separation of concerns

• Clear interfaces

• Independent modules

15. Future Enhancements

• Quiz Generator Agent

• Personalized Learning Agent

• Vector memory integration

• Cloud deployment

• Multi-user access

16. References

The system design is informed by established research and literature in artificial intelligence, multi-agent systems, large language models, and AI-driven educational technologies. The architectural decisions, agent coordination mechanisms, and structured reasoning workflow are inspired by foundational and contemporary research in intelligent agents and autonomous planning systems.

• *Artificial Intelligence: A Modern Approach,* Stuart Russell & Peter Norvig

• Research on intelligent agents and deliberative planning systems

• Academic studies on Agentic AI architectures and multi-agent collaboration frameworks

• Research on Large Language Models (LLMs) and tool-augmented reasoning systems

• Literature on AI-driven educational systems and intelligent tutoring platforms

• Studies on modular software architecture and distributed AI systems

• Research on retrieval-augmented generation (RAG) and knowledge-grounded AI systems

17. Conclusion

The Multi-Agent Education System successfully implements Agentic AI principles by distributing responsibilities among specialized agents and coordinating them through structured orchestration. The system ensures modularity, scalability, structured reasoning, and high-quality educational output generation. Its layered architecture, controlled workflow, and modular agent design make it suitable for academic submission and internship evaluation.