ChatGPT

Agentic AI Orchestration Pipeline for Educational Content

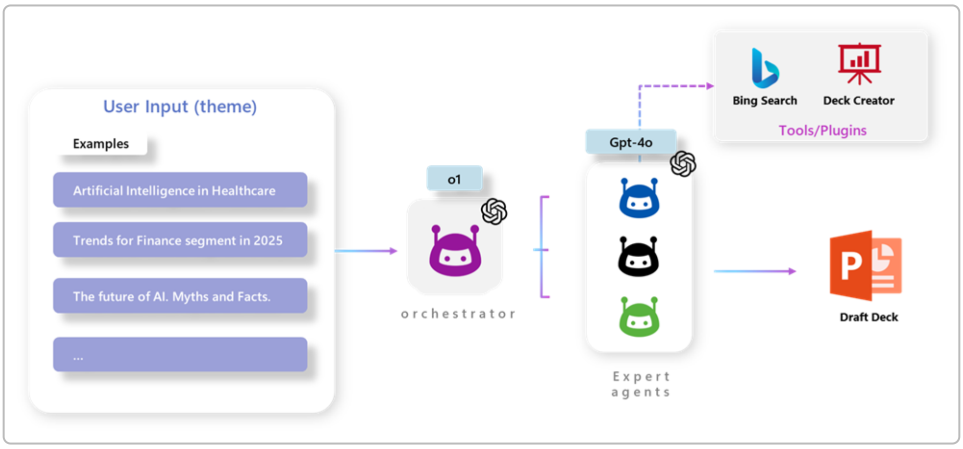

Figure: Illustration of a multi-agent orchestration pipeline, where a central coordinator (orchestrator) spawns specialized AI agents to collaboratively generate educational materials. In this system, **Agentic AI orchestration** coordinates multiple AI "expert" agents to work together on a shared goal [1] [2]. Instead of one monolithic model, each agent has a distinct role (for example, generating content, creating worksheets, or producing art). An orchestrator agent (the **Sahayak Coordinator**) manages the workflow: it routes inputs to the right agents, enforces permissions, and aggregates their outputs. Each agent acts semi-autonomously, making context-aware decisions and calling external tools as needed [1] [2]. This multi-agent approach (often called the *multi-agent collaboration* paradigm) allows complex educational tasks to be broken into sub-tasks that different agents handle in parallel [1] [2].

Input Handling and Agent Dispatching

Figure: AI-driven classroom setting (illustration). The pipeline begins when a teacher provides input. This can be a **text prompt** (e.g. a question or instruction typed into chat) or a **visual input** (e.g. a photo or PDF of a textbook page). A preprocessing step first examines the input type. If it's an **image or PDF page**, an OCR or vision tool (as an external service) extracts the text. The orchestrator then dispatches tasks as follows: - **Image/PDF input**: routed to the **WorksheetAgent**. This agent takes the extracted text and generates instructional materials (slides, differentiated worksheets, etc.). It may also spawn other agents (see below).
- **Text/Chat input**: routed to the **ContentAgent**. This agent treats the prompt as a direct question or instruction and generates explanatory content or lesson material in response.

For example, if a teacher uploads a page from a science textbook, the system first applies OCR to get the text. Then the WorksheetAgent uses that text to create slides and practice worksheets. If the teacher instead types "Explain photosynthesis," the ContentAgent is triggered to produce a coherent explanation. This initial routing ensures each agent only handles inputs of the type it is designed for. (In implementation, an OCR service itself can be exposed as an MCP *tool/server* that agents can call when needed.)

ContentAgent (Concept & Lesson Generation)

The **ContentAgent** specializes in generating instructional text (lectures, explanations, examples) from a teacher's query. It uses a large language model (LLM) to craft coherent lessons or explanations on demand. For instance, given "Describe the water cycle," the ContentAgent would generate a structured answer covering evaporation, condensation, etc., complete with analogies or examples. It can also build outlines or lesson plans. In modern systems, general LLMs like ChatGPT are often used for this role: such models can produce detailed lesson plans, draft instructional materials, and even generate assessments based on a topic [3] [4]. For example, ChatGPT is widely used to generate educational content across subjects, showing its versatility in creating explanations and even quiz questions [4].

After producing text, the ContentAgent can optionally trigger other agents. For instance, if the teacher's prompt implies deeper learning ("Explain X in detail"), the ContentAgent might call the **ExamPaperAgent** to formulate practice questions. Conversely, for a straightforward Q&A, the ContentAgent's response is delivered directly (with no exam agent needed). Throughout, the ContentAgent can also integrate tool use: for example, calling a web search or database tool (via MCP) if up-to-date information is needed.

WorksheetAgent and ArtAgent (Materials & Visuals)

The **WorksheetAgent** takes textbook content (from an image/PDF) or high-level lesson content (from the ContentAgent) and turns it into student-ready materials: slides (PPT) and differentiated worksheets. Its tasks include summarizing key points into slide decks and formulating exercises at varying difficulty levels (easy/medium/hard) to accommodate different learners. This mirrors features in AI curriculum tools, which align content to educational standards and support *differentiated instruction* [5]. For example, MagicSchool AI automatically generates lesson plans aligned with standards and "supports differentiated instruction to accommodate diverse student needs" [5]. Our WorksheetAgent similarly creates multiple versions of a worksheet or exercise set (e.g. simpler and advanced questions) so that all students can engage at their level.

Crucially, the WorksheetAgent also invokes the **ArtAgent** to generate visual aids. Whenever a slide or worksheet would benefit from an illustration (e.g. a science diagram, chart, or cover image), the WorksheetAgent calls the ArtAgent as an external tool. The ArtAgent uses image generation models (like DALL·E or Stable Diffusion) to create custom visuals. For instance, if a worksheet is about plant biology, the ArtAgent might draw a labeled plant diagram. Many AI education platforms include such image features: for example, Teachfloor offers "AI Cover Images" and "AI Image Generation" to create custom lesson visuals [6]. By delegating visuals to a dedicated agent, we ensure high-quality, contextually relevant images without bloating the text generation agent.

The output of the WorksheetAgent (text + images) is assembled into presentation slides and worksheet documents. These can be exported as PPTX files, PDFs, etc., ready for the teacher to review or distribute.

QA-Agent and ExamPaperAgent (Questioning and Assessment)

A **QA-Agent** can handle on-the-fly questions that don't require full content generation. If a teacher or student asks a simple factual question ("What is the capital of France?") or follows up on the generated materials, the QA-Agent can use retrieval or LLM answers to respond. This agent acts like a "chatbot tutor" for quick queries, separate from the main content pipeline.

The **ExamPaperAgent** is triggered when comprehensive assessment questions are desired. We call this agent only if the context warrants it (e.g. the teacher asks to generate exam-style questions, or if a conceptual explanation is provided and we want practice problems). For example, after the ContentAgent explains a topic, the system might ask "Create practice exam questions on this topic." The ExamPaperAgent then uses the LLM (and possibly structured question-generation templates) to produce high-quality questions (multiple-choice, short-answer, etc.) covering all subtopics. This is analogous to AI quiz generators in many platforms: as one AI curriculum tool notes, its "AI Quiz Generator" can create structured quizzes in seconds and "AI Assessment Builder" can design evaluation activities based on course content [3]. Thus our ExamPaperAgent ensures that teachers get a bank of practice questions or an exam paper aligned with the material.

When to trigger ExamPaperAgent: If the teacher's input is a request for an explanation or broad concept discussion (e.g. "Explain these concepts"), we generate that content and then invoke the ExamPaperAgent to produce practice questions. If the teacher's input is a narrow factual question, we answer directly without generating an exam. This conditional triggering ensures efficiency and relevance.

MCP-Based Orchestration Architecture

To coordinate all agents and external tools, we use the **Model Context Protocol (MCP)** as an orchestration framework. MCP defines a standardized **host–client–server** architecture for AI agents [7] [8]. In our design, the **Sahayak Coordinator** (a cloud function or similar service) serves as the **MCP Host** [7]. It implements the orchestration logic: receiving the teacher's input, spawning agent instances, and enforcing permissions. Each specialized agent (ContentAgent, WorksheetAgent, ArtAgent, QA-Agent, ExamPaperAgent) includes an **MCP Client** component [9]. The client converts the agent's requests into the MCP protocol and communicates with remote services.

External services (data sources and tools) are exposed as **MCP Servers**. For example, a Firestore-based database can run as an MCP Server to provide persistent memory (storing teacher profiles, past content, or curricular standards). Likewise, OCR, web search, or any API can be wrapped in an MCP server. These servers listen via streamable HTTP transports [10] so that multiple agent clients can connect concurrently and maintain stateful sessions. Cloud Run is an ideal hosting platform for MCP servers: it easily scales multiple server instances and supports server-sent events (SSE) streams for bidirectional communication [10].

In this MCP setup, when an agent (client) needs an external resource, it issues an MCP call. For instance, the WorksheetAgent's client might call the OCR server, or the ArtAgent's client calls an image-generation server. The host (Coordinator) can route these calls and ensure only authorized agents use each service. Overall, MCP provides a unified "plug-and-play" interface for our agents to access tools without custom integration code [8].

Personalization and Curriculum Alignment

An important feature of the system is **personalization** for the teacher and curriculum. The orchestrator can query the teacher's profile or preferences at the start. For example, the teacher might indicate their preferred **language** (e.g. Hindi) or which **curriculum/standards** they follow (e.g. CBSE, state board, IB). The agents then tailor all outputs accordingly. Modern LLMs (like GPT-4) support multilingual text generation [11], so if a teacher prefers Hindi, the ContentAgent and other agents simply generate content in Hindi. This ensures the material is immediately usable in the teacher's context.

Similarly, curriculum alignment can be handled. The system might ask "Which syllabus or standards does this lesson follow?" and then constrain content to that framework. For instance, MagicSchool AI explicitly aligns content to K–12 standards and provides "customizable, differentiated instruction" based on them [5]. Our agents can do likewise: if the teacher says they follow a particular syllabus, the WorksheetAgent and ContentAgent can incorporate that scope (for example, focusing on chapters relevant to the syllabus). Personalization also extends to teaching style: if a teacher wants more visual content, the ArtAgent can be emphasized; if they want more examples vs. theory, the ContentAgent can adjust. Overall, AI enables *precision teaching* where the pipeline adapts to the user's needs. As one review notes, AI tools can automate routine tasks and provide insights to help teachers refine and personalize their teaching strategies [12].

Key personalization points: Agents can store user preferences in memory (via an MCP server), and generate language- or curriculum-specific variants of every output [5] [11]. Over time, the system learns a teacher's habits (e.g. always output in Hindi) and applies them automatically, making the content pipeline feel tailored to each educator.

Summary of the pipeline: In practice, this orchestrated pipeline might work as follows:

1. **Teacher input** (image or text) → Orchestrator.
2. **Orchestrator** decides input type.
3. If **image/PDF**: call OCR (MCP tool), then send text to WorksheetAgent.
4. If **text/chat**: send prompt to ContentAgent.
5. **WorksheetAgent** (given textbook text or content): create slide decks and worksheets (with varied difficulty), and call ArtAgent for images [6].
6. **ContentAgent** (given prompt): generate lesson/explanation. If needed, trigger ExamPaperAgent.
7. **ArtAgent**: receive prompts (e.g. "diagram of plant cell") and produce illustrative images.
8. **ExamPaperAgent**: generate exam questions when triggered (e.g. after conceptual content).
9. **QA-Agent**: handle direct Q&A or fact lookup on request.
10. **MCP servers**: throughout, provide memory (preferences, content history), search or database access as needed [8] [10].

11. **Output assembly**: Combine texts and images into PPT and worksheet documents, finalize exam question sets, and present these to the teacher for review.

All components work together in a closed loop. This agentic orchestration ensures that from a single teacher request, the appropriate combination of content, assessment, and visual materials is produced automatically. The pipeline leverages MCP to plug in any new tools or data sources as needed (for example, a translation server or a subject-specific knowledge base), making it flexible and extensible.

Sources: This pipeline design follows principles of multi-agent AI systems in education [2] [1], combined with MCP for tool integration [7] [8]. The idea of specialized agents generating curricula, quizzes, and images is supported by existing AI educational tools [3] [6]. Personalization of content (language, curriculum) is a well-recognized advantage of AI teaching assistants [5] [11]. All components above are grounded in current research and platforms for agentic AI orchestration.

---

[1] Agentic AI Orchestration: Definitions, Use Cases & Software
https://www.warmly.ai/p/blog/agentic-ai-orchestration

[2] AI Agents in Education
https://arxiv.org/pdf/2504.20082

[3] [4] [5] [6] 9 Best AI Course Curriculum Generators for Educators [2025] - Teachfloor Blog
https://www.teachfloor.com/blog/ai-curriculum-generator

[7] [8] [9] What is Model Context Protocol (MCP)? | IBM
https://www.ibm.com/think/topics/model-context-protocol

[10] Transports - Model Context Protocol
https://modelcontextprotocol.io/docs/concepts/transports

[11] Assessing Generative AI In Multilingual Content Creation
https://identrics.ai/blog/generative-ai-multilingual-content-creation/

[12] Personalized Learning Platforms In 2025: AI In Education - eLearning Industry
https://elearningindustry.com/ai-in-education-personalized-learning-platforms