The Problem Statement and Project Blueprint

Learn how to design a multimodal AI assistant by creating an architectural blueprint that orchestrates vision and research servers using MCP.

Our agent has successfully mastered text-based tools, skillfully interacting with web APIs and private knowledge bases to perform complex tasks. However, a vast amount of information in our world is visual, locked away in images and diagrams. In this module, we will cross that frontier by building our first multimodal application: an intelligent "Image Research Assistant" that demonstrates how MCP can orchestrate completely different types of intelligence, i.e., vision and text retrieval, to solve a single, complex problem.

The problem statement

Imagine a researcher is browsing a digital archive and finds an intriguing photograph of a grand, historic building. The image is captivating, but it lacks any context. There's no caption, no metadata, nothing to identify the structure or its location. The researcher is left with several fundamental questions: What is this building? Where is it located? And what is its historical significance?

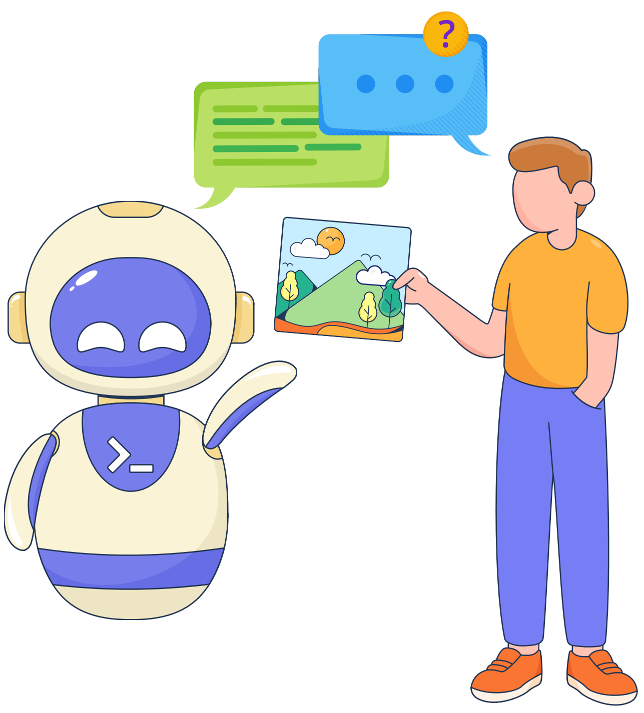
Ask

To answer these questions, the researcher would have to embark on a disjointed, manual workflow. First, they might use a reverse-image search tool to hopefully identify the landmark. Then, armed with a name, they would switch to a new browser tab to search Wikipedia or a search engine for articles about its history. This multi-step, copy-and-paste process is inefficient and highlights a key challenge of seamlessly bridging the gap between visual identification and deep knowledge retrieval.

The solution: An “Image Research Assistant”

Our solution is to build an intelligent, automated “Image Research Assistant.” This assistant will replace the manual, multi-step workflow with a single, seamless interaction. A user will simply upload the unidentified image, and our agent will orchestrate a complete research process, delivering a final summary with relevant articles and links.

Ask

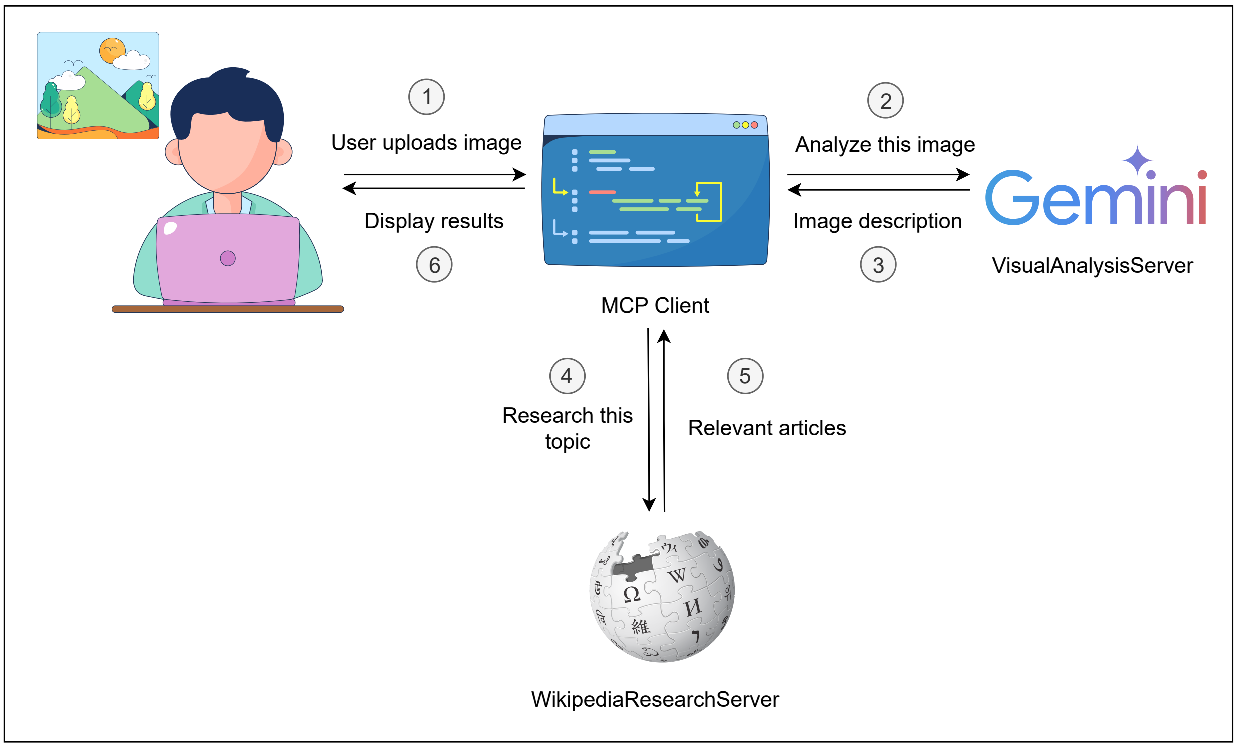


The key to this solution lies in its agentic architecture, orchestrated using the MCP. Instead of building one monolithic application, we will create two *highly* specialized, single-purpose servers. The agent's "brain" will act as the MCP client, first calling our vision server to understand the image and then using that output to call our research server. By decoupling these distinct capabilities, we create a system that is not only powerful but also modular and extensible.

Architectural design of the system

To achieve this automated workflow, we will design our system around a central orchestrator, our MCP client, which will act as the agent's brain. This agent will execute a clear, two-step plan. First, it will dispatch the user's uploaded image to a specialized `VisualAnalysisServer` to get a text description. Second, once it receives that description, it will immediately use the text to query an independent `WikipediaResearchServer` for relevant articles. This entire orchestration, from the data handoff to the invocation of distinct tools on different servers, is managed seamlessly through the MCP. This decoupled architecture, where each component has a single, well-defined responsibility, is the key to building a scalable and maintainable multimodal system.

Ask



Architectural design of the “Image Reseach Assistant”

The MCP servers: Specialized workers

Our system’s power comes from two specialized, single-purpose MCP servers, each handling a distinct part of the research process. By keeping their responsibilities separate, we create a modular and reusable architecture.

- The `VisualAnalysisServer`: This server will be our computer vision expert. Its sole purpose is to receive an image and return a rich, textual description of its contents. To achieve this, we will implement a tool that uses a powerful Gemini vision model, which can analyze the image’s content and identify specific objects or landmarks within it.
- The `WikipediaResearchServer`: This server will act as our dedicated research librarian. It will expose a tool that leverages a standard Python `wikipedia` library to perform its task. This tool will take a text query (the description provided by our `VisualAnalysisServer`) and return a list of the most relevant Wikipedia articles, complete with summaries and URLs.

The MCP client: The brain of the assistant

Ask

The client is the central hub of our entire application; it is the orchestrator that executes the research plan. Its job is to manage the flow of information between the UI and our specialized servers. Building this kind of intelligent, multi-step agent might seem complex, but we have already built the perfect foundation for it. For this role, we will reuse the same LangGraph-based client from our previous lessons. Its state-machine architecture is perfectly suited for managing this kind of sequential, multi-tool workflow where the output of one tool becomes the input for the next.

UI of the application

To make our research assistant accessible and easy to use, we will build a simple but effective web-based user interface using **Gradio**. This popular Python library is specifically designed for creating interactive demos for machine learning models with minimal code. It allows us to quickly wrap our agent's logic in a user-friendly interface, letting us focus on the orchestration rather than on complex front-end development.

Our Gradio application will have two main functions. It will provide an upload widget that allows the user to easily select an image file from their computer, and it will feature a clean display area where we will present the final results

Ask