Part 1 – Reading and Exploration

The Model Context Protocol (MCP) is a new system that helps AI programs connect to outside tools and data in a simple and standard way. Before MCP, every AI agent or chatbot needed special code to use each different API or service. This made things hard to manage. MCP was created to fix that problem. It acts like a universal connector that allows any AI agent to use any compatible tool without needing to be rebuilt each time.

According to the article on Hugging Face, MCP uses a client and server structure. The server offers different "tools" or actions, like searching for a place or looking up data. The client (for example, an AI agent) connects to the server and asks what tools are available. The server then sends back a list of tools, along with clear instructions on what inputs are needed and what kind of data it will return. This information is shared in a structured JSON format, which makes it easy for computers to read and understand. MCP also supports different ways to connect, like HTTP or standard input/output (stdio).

The goal of MCP is to make AI agents more flexible, modular, and interoperable. That means agents can easily use new tools, combine different data sources, and perform real-world actions, like checking maps, weather, or databases, without needing special coding each time. It also makes it easier for developers to build and share reusable tools that any AI model can use safely.

When I looked at existing map servers, I saw several smart design patterns. OpenStreetMap (OSM) provides free and open map data that anyone can use. It offers geocoding (turning a place name into coordinates) and reverse geocoding (turning coordinates into a place name). MapLibre and Leaflet focus on showing maps visually in web apps. They use small images called map tiles, which are requested using URLs like /z/x/y for zoom and position. These systems are modular, simple, and cache-friendly.

Another example, OpenRouteService, provides tools for routing, distance calculation, and travel-time maps (isochrones). Many of these APIs use clear JSON data formats, have rate limits, and expect polite usage with a custom *User-Agent* header.

From these examples, I learned that good map servers share some key ideas:
they use simple structures, keep their tools separate, respond quickly, and organize their data clearly. These same ideas also guide MCP, it encourages clean communication between AI agents and tools, helping them work together easily and efficiently.