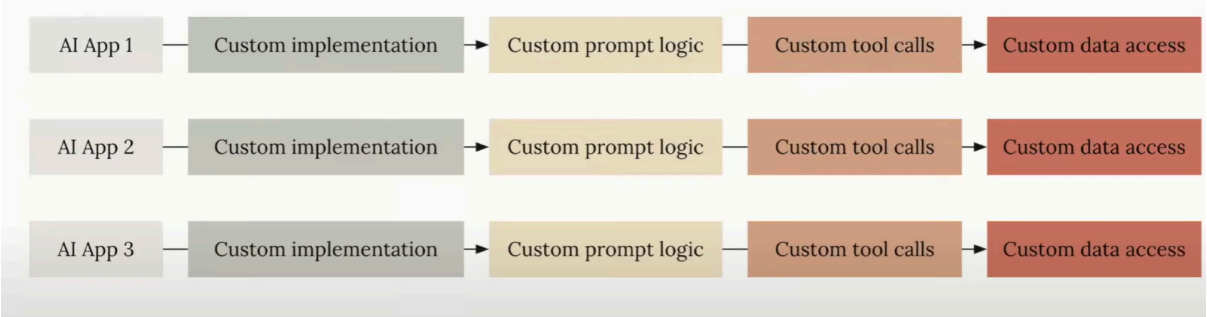
 # Model Context Protocol (MCP) Report

Overview
The **Model Context Protocol** (MCP) is an **open** protocol designed to enable **seamless integration** between Large Language Model (**LLM**) applications and external **data sources** and **tools**. It serves as a **standardized way** to connect LLMs with the **context** they need to perform tasks effectively.

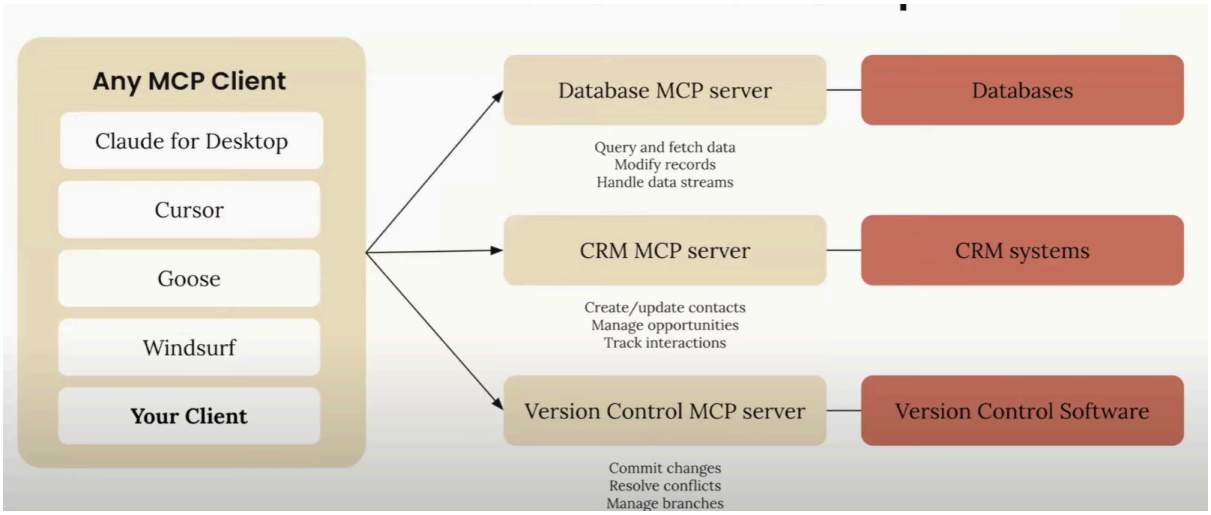
Problem Statement
Prior to MCP, AI systems faced challenges with:
- Fragmented integrations between LLMs and various data sources
- AI models being isolated from necessary data
- Information trapped in silos
- Multiple custom implementations required for each integration

Solution
MCP replaces these multiple custom implementations with a single universal protocol, creating a standardized method for connecting AI systems with external data and tools.

Without MCP →



With MCP →



For enterprises mainly, with the help of MCP, concerns can be separated. For instance, a Data Infrastructure team working on Vector DB/RAG can now provide access to other teams building AI apps. Previously, every other team would build their own system of accessing database, chunking, prompting and so on. With MCP, an enterprise can have a team that owns the Vector DB interface which can be turned into an MCP Server – which can be centralised!!

Importance
MCP provides significant advantages as a:
- Universal, open standard
- Simpler and more reliable method to give AI systems access to necessary data
- Framework that enables better scaling of connected systems

Use Cases
MCP can be effectively utilized in various scenarios including:
- Building AI-powered Integrated Development Environments (IDEs)
- Enhancing chat interfaces with contextual data
- Creating custom AI workflows
- Connecting AI systems with external data sources

Key Advantage
The primary advantage of MCP is that it eliminates the need for custom implementations for each new data source by providing a single, standardized protocol. This makes it easier to scale and maintain AI systems that need to access multiple data sources.

How MCP Works
MCP servers provide a way for AI models to access the context needed to accomplish tasks. The protocol defines how clients (like LLM interfaces) can request and receive information from servers.
MCP servers can provide four types of context primitives to AI models:

1. **Tools**: Functions that the model can use to perform actions like creating or updating a database
   - Example: Checking the sum of orders in a SQLite database

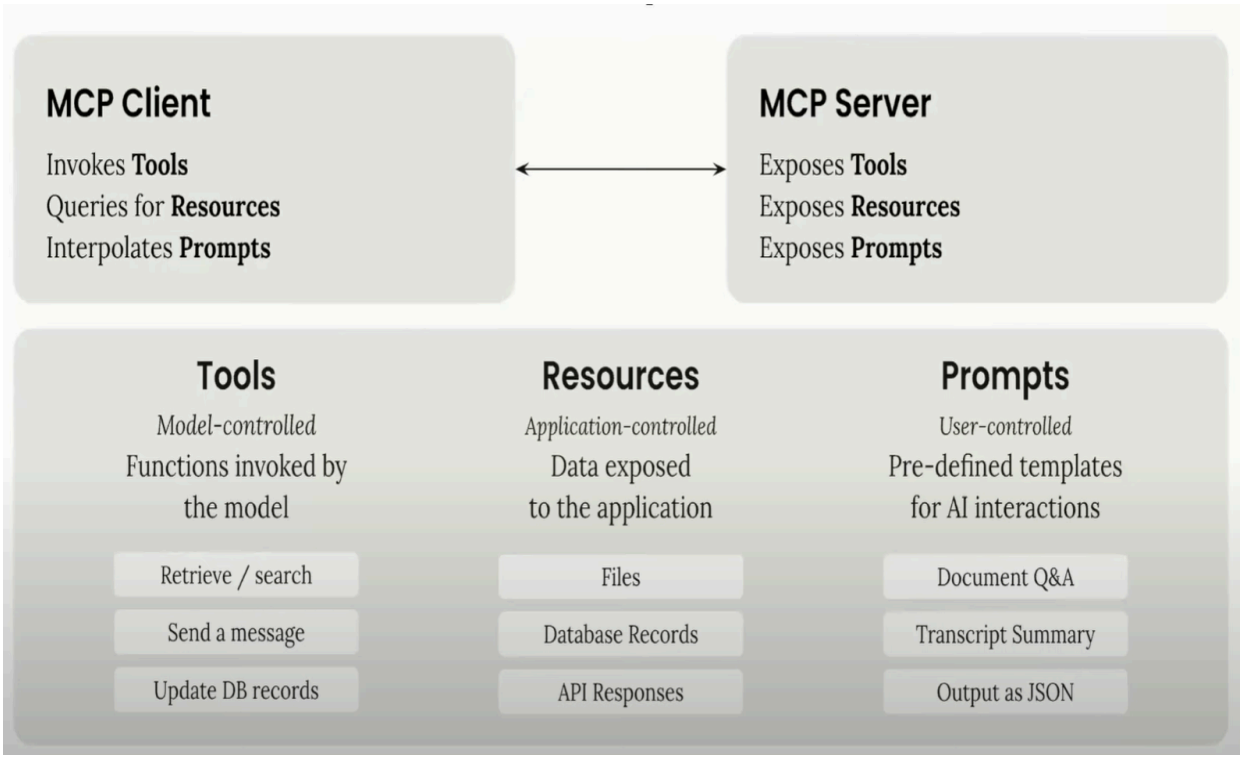
2. **Resources**: Attachments provided to a model
   - Example: Accessing presentation files for verification

3. **Sampling** (less commonly used): A way for the model to query other models

4. **Prompts** (less commonly used): Parameterized prompts that clients can use as templates when making requests to the model

MCP Architecture
- **MCP Servers**: Programs that handle requests from MCP clients. They implement the tools or provide the resources.

-



MCP Clients: Make requests for tools/resources (e.g., Chat interface of Claude 3.7 Sonnet)

Example Scenario
```
LLM --- MCP server ---> SQLite Database
LLM --- MCP Server ---> Presentation (Present locally)
```

In this scenario, Claude (the LLM) can:
1. Connect to an MCP server that provides access to a SQLite database
2. Request information about the sum of all orders in the database
3. Connect to another MCP server (or the same one with different capabilities) to access presentation files
4. Compare the data between these sources to verify consistency

This allows the AI to seamlessly work with external data sources without requiring custom integration for each source.