AIMCP - MCP Server

# 1. Introduction

The AIMCP project is an implementation of a Model Context Protocol (MCP) server. It enables client applications to interact with multiple Large Language Models (LLMs) and tools in a standardized way. The current implementation primarily integrates with the Mistral 7B Instruct model (mistral:7b-instruct-v0.3-q8\_0) via the Ollama runtime.

# 2. What is an MCP Server?

The Model Context Protocol (MCP) is an emerging protocol that defines a standardized way for clients and servers to communicate with LLMs and tools. An MCP server acts as a bridge between clients (such as a ChatGPT-like UI) and resources/tools (such as LLMs, databases, knowledge stores, or APIs).

Key responsibilities of an MCP server include:

• Handling WebSocket/RPC communication with clients

• Managing and exposing resources

• Interfacing with one or more LLMs or tools

• Returning results in MCP-compatible JSON-RPC formats

# 3. AIMCP Architecture

The architecture of AIMCP consists of the following components:

• MCPWebSocketHandler – Manages WebSocket connections, request parsing, and response dispatch.

• OllamaLlmAdapter – Connects to the Ollama runtime for interacting with the Mistral model.

• MCPResourceService – Provides resource discovery and management APIs.

• UI (ChatGPT-like interface) – Frontend for streaming responses and user interaction.

These components together allow clients to connect to the MCP server, discover resources, and query the Mistral model.

# 4. Technology Stack

• Java (Spring Boot) – Core server implementation

• WebSockets + JSON-RPC – Communication protocol

• Ollama – Runtime environment for running and serving Mistral LLMs

• Mistral 7B Instruct v0.3 (q8\_0) – Current default LLM

• HTML/CSS/JS – UI layer mimicking ChatGPT

# 5. Current Capabilities

• Works as an MCP-compatible server with WebSocket connections.

• Integrates with Mistral model through Ollama.

• Provides resource discovery and structured JSON-RPC responses.

• Frontend supports ChatGPT-like interface with streaming responses.

# 6. Advantages

• Standards-based MCP communication model.

• Decoupled LLM adapter allows extension with multiple LLMs.

• Simple ChatGPT-style UI supports user-friendly interaction.

• Supports streaming partial responses.

# 7. Current Limitations

• Only one LLM (Mistral) integrated – lacks multi-model support.

• No enterprise-grade observability, logging, or monitoring yet.

• No authentication, RBAC, or encryption mechanisms currently.

• Resource and tool management still basic.

# 8. Enhancements Needed for Enterprise-Grade MCP Server

• Multi-LLM integration (Falcon, Llama, GPT family, etc.)

• Pluggable tool ecosystem (search engines, vector databases, APIs).

• Security features: OAuth2, RBAC, TLS, audit logging.

• Observability: Metrics, tracing, structured logging.

• Scalability: Load balancing, distributed deployments.

# 9. Hosting Options

* On-Prem Deployment:

• Run Java Spring Boot service locally or in private data centers.

• Deploy Ollama on dedicated GPU nodes for inference.

* Cloud Deployment:

• Containerize with Docker/Kubernetes.

• Host on AWS ECS/EKS, Azure AKS, or GCP GKE.

• Use managed databases and observability stacks.

# 10. Mistral LLM in AIMCP

The current implementation uses the Mistral 7B Instruct v0.3 model in quantized q8\_0 format. It is served via the Ollama runtime locally. This LLM is capable of handling instruction-following tasks and provides fast inference for prototyping MCP-based applications.

# 11. Adding More Tools

MCP servers are designed to expose multiple tools. In AIMCP, new tools can be added by creating adapters similar to the OllamaLlmAdapter. Example tools include:

• Vector Database Adapter (e.g., Pinecone, Weaviate) – For RAG workflows.

• Search Engine Adapter (e.g., Bing, Google APIs).

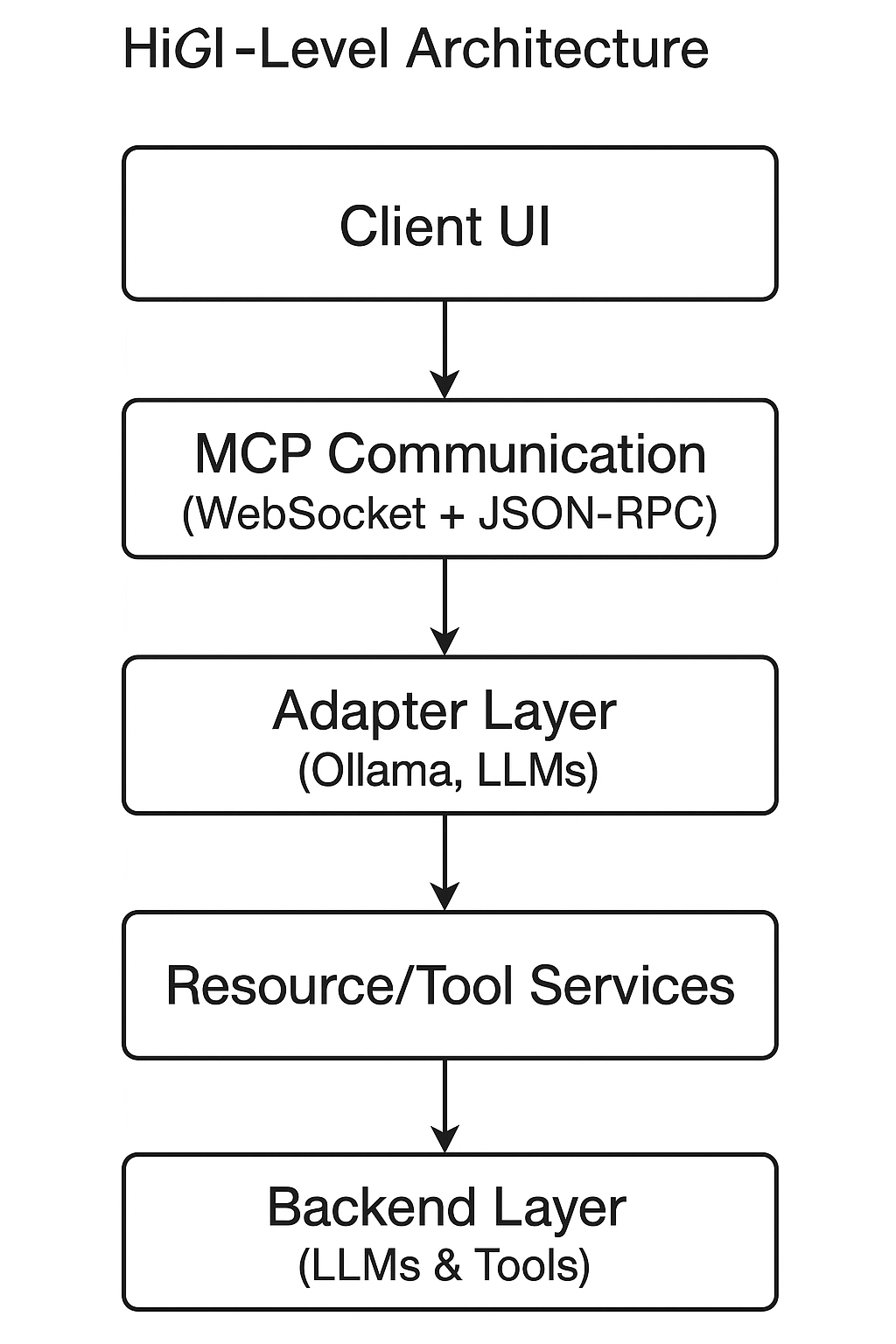
• Knowledge Base Adapter (e.g., company policies, documents).

• External API Connectors (e.g., Jira, Salesforce).

## ****12. Sequence Flow – Chat Message****

**Step-by-Step Flow:**

1. Client types message → sends JSON-RPC request over WebSocket.
2. MCPWebSocketHandler receives request → parses it.
3. Server checks request type: **LLM query** or **tool call**.
4. For LLM query: request forwarded to OllamaLlmAdapter.
5. Adapter calls Ollama REST API with Mistral prompt.
6. Ollama streams response → Adapter formats it as MCP response.
7. MCP server streams response back to UI.
8. Client UI renders typing effect.



# 13. Conclusion

AIMCP is a functioning prototype of an MCP server with integration to Mistral LLM. While it demonstrates core MCP capabilities (resources, tools, JSON-RPC handling, streaming UI), it requires significant enhancements to be considered enterprise-grade. With security, observability, scalability, and multi-LLM/tool support, it can evolve into a robust enterprise-grade MCP server.