Decision Document: MCP Tools Deployment Strategy

Template for United States of America  
Fixed-Price Standalone SOW

July 4, 2025

Presented By:

AI POC Team

Technical Lead

[Office Address]

[Office Phone Number]

[Email]

Revision Table

|  |  |  |
| --- | --- | --- |
| Revision | Date | Description of Change |
| 1.0 | XXXXX | Initial Release |

*Confidentiality Statement*

*This document contains information that is privileged, confidential, and otherwise protected from disclosure to anyone other than intended recipient(s). As a result, this information should not be disclosed, duplicated or used for any other purpose than to evaluate Actalent. On signing the contract as a result of the process initialized by submission of this document, any right to disclose, duplicate or use the data contained within this document will be to the extent provided in the resulting agreement*

Table of Contents

Table of Exhibits

[Exhibit A – Change Order 27](#_Toc102744285)

# Decision Document: MCP Tools Deployment and Development Strategy

## Title and Metadata

|  |  |
| --- | --- |
| **Field** | **Value** |
| Decision Title | MCP Tools Deployment Strategy and Development Freeze |
| Decision Number | DDR-001 |
| Status | Accepted |
| Proposed By | AI POC Team |
| Teams Involved | Technical Development, Operations, Security |
| Date | July 4, 2025 |
| Tags | Development, Infrastructure, Security |

## Background/Context

The MCP (Model Context Protocol) tools for Word document generation have been under development for 4 weeks, reaching approximately 60% completion of the major work. This project aims to automate technical document generation and Statement of Work (SOW) creation using Large Language Models (LLMs) integrated with Microsoft Word through specialized tools.

The current implementation successfully integrates with GitHub Copilot extension using Claude Sonnet AI models for long-form document generation. The system demonstrates significant productivity improvements in content generation phases, though validation and review processes require additional time investment.

Key contextual factors driving this decision include the need for deployment strategy clarity, security considerations, and development resource allocation. The remaining 30% of unidentified work presents unknown challenges that may surface during production usage.

## Alternatives Considered

### Deployment Options

|  |  |  |
| --- | --- | --- |
| **Option** | **Pros** | **Cons** |
| Local deployment with Python | • No operational costs • Direct hardware control • Requires Python environment | • Security concerns with direct system access • Limited scalability • Local resource constraints |
| Cloud server deployment | • Better scalability • Professional hosting environment • Centralized management | • Operational costs required • Need for SSE and streamable HTTP expertise • Custom interface development needed |

### LLM Model Selection

|  |  |  |
| --- | --- | --- |
| **Model Type** | **Advantages** | **Disadvantages** |
| Closed-source models (Claude Sonnet, GPT) | • High performance and accuracy • Proven benchmarks • Enterprise support | • Data sent outside organization • Subscription costs • NDA compliance concerns |
| Open-source models (LLama 4, Mistral,Qwen) | • Data retained locally • Customizable through fine-tuning • No ongoing subscription fees | • High capital investment for hardware • Maintenance team required • Moderate performance compared to closed-source |

### Development Approach Options

1. Continue development to complete remaining 30% before deployment
2. Freeze current development at 60% completion and address issues as they arise
3. Complete overhaul with focus on addressing known architectural issues

## The Decision

After careful evaluation of all alternatives, the following decisions have been made:

### Development Freeze at 60% Completion

We will freeze the current codebase at 60% completion and address the remaining 30% of unidentified work when the backlog of issues reaches 10 items. This approach allows for immediate deployment and real-world testing while maintaining development efficiency.

### Local Deployment with Docker Containerization

To mitigate security concerns, the MCP tools will be deployed locally using Docker containers with access restricted to mounted folders only. This provides the cost benefits of local deployment while implementing necessary security boundaries.

### Continue with Claude Sonnet for Production

Given the proven performance of Claude Sonnet with GitHub Copilot extension for long-form document generation, we will continue using this model combination while evaluating open-source alternatives like LLama 4 for future consideration.

## Impact

### Positive Impacts

* Significant productivity improvements: 4x faster content generation phase
* Overall time savings of 37.5% for document creation workflows
* Faster time-to-market for the MCP tools solution
* Reduced operational costs through local deployment
* Enhanced security through Docker containerization

### Trade-offs and Challenges

* Validation phase requires 75% more time investment (35 minutes vs 20 minutes manual)
* 30% of functionality remains untested and may require future development sprints
* Potential system hallucinations requiring AI response evaluation or human oversight
* Unreliable tool usage may occur, requiring improved tool definitions and annotations
* Limited scalability with local deployment compared to cloud solutions

### Quantitative Impact Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Metric** | **Manual** | **AI-Assisted** | **Improvement** | **Productivity Ratio** |
| Generation Phase | 60 minutes | 15 minutes | +75% time saved | 4.0x faster |
| Validation Phase | 20 minutes | 35 minutes | -75% time increase | 0.57x (slower) |
| Overall Workflow | 80 minutes | 50 minutes | +37.5% time saved | 1.6x faster |

## Optimization Strategies

To address identified challenges and improve system reliability, the following optimization strategies will be implemented:

1. Implement AI Agent Response Evaluation (AI Judge) to reduce hallucinations
2. Enhance tool definitions with comprehensive annotations and parameter descriptions
3. Apply advanced prompting techniques including Chain-of-Thought and Active-Prompt strategies
4. Develop multi-agent system architecture to handle complex tool interactions
5. Integrate Corrective RAG and Hybrid Search for improved information retrieval or CAG.

## Technical Considerations

### Token Economics and Cost Management

LLM billing operates on token-based pricing models (approximately $0.01 per million tokens). Cost optimization requires careful management of input tokens (prompts + context + history) and output tokens (model responses). The Claude Sonnet model's 120k token context window provides sufficient capacity for most document generation tasks.

### Role-Based Prompt Architecture

|  |  |
| --- | --- |
| **Role Type** | **Purpose and Functionality** |
| System Role | Sets context, guidelines, behavioral constraints, and response style |
| User Role | Contains specific questions, tasks, contextual information, and examples |
| Assistant Role | Designates the LLM's structured response output |

## References

1. Qiao et al. (2023) - Reasoning Enhancement in Large Language Models
2. Claude Sonnet API Documentation - Context Window and Token Management
3. GitHub Copilot Extension Integration Guidelines
4. Docker Security Best Practices for AI Applications
5. LLM Benchmarking Standards and Performance Metrics
6. MCP (Model Context Protocol) Technical Specifications
7. AI Agent Response Evaluation Methodologies

## Next Steps and Action Items

1. Set up Docker containerization environment for local deployment
2. Establish issue tracking system with 10-item threshold for development resumption
3. Implement comprehensive tool definition annotations
4. Begin evaluation of open-source LLM alternatives (LLama 4 Scout)
5. Design AI response evaluation framework
6. Create user training materials for validation process optimization
7. Establish monitoring and logging for production deployment