AI Task Agent

Team Member Contacts

Jadon Duff, Computer Science (jduff2024@my.fit.edu)

Jacob Ewasko, Computer Science (jewasko2024@my.fit.edu)

Jacob Lebkuecher, Software Engineering (jlebkuecher2023@my.fit.edu)

Ioannis Limniatis, Computer Science (ilimniatis2023@my.fit.edu)

Faculty Advisor

Eraldo Ribeiro, College of Engineering and Science: Department of Electrical Engineering and Computer Science (eribeiro@fit.edu)

- **Advisor Meetings:** 8/18/2026, 8/19/2026, 8/20/2026, 8/25/2026

Client

Doug Armstrong (District Directorate Chief – Response, US Coast Guard Auxiliary)

- **Client Meetings:** 8/26/2026

Goal and Motivation

Explain project goals and overall motivation.

In our everyday life, we frequently do repetitive actions on our computers, such as checking our email, managing calendars, and browsing the internet for the same websites over and over. These repetitive actions on computers have become essential to our everyday life, but the repetitive actions become time consuming. Even though that time can be better spent elsewhere, we are required to invest our time in them because of how essential they are to our jobs, social life, and communications. These repetitive tasks can be no longer a waste of your time by using our AI Task Agent to accomplish them.

There are many sectors in our technological world that can be enhanced to provide more value. Currently, MCP servers are tuned to a single domain and preprogrammed tools. This only lets the server do specified actions for an entity, however as a result it becomes not very flexible at all. LLMs are also constrained by their cutoff window (memory) and tuned for general versatility. This constrained cutoff window is sometimes too large for a simple task and sometimes not enough for larger, more complex tasks. Since LLMs are tuned for general versatility, they struggle with specific actions as a side effect. Finally, a multitude of systems and APIs can lead to security concerns and vulnerabilities, risking data theft and loss. While some of these individual subproblems have been attempted in industry, they have not been solved altogether, where our solution to all of these technologies working together is our AI Task Agent.

Approach

Three key features from a user-oriented approach.

Feature 1 (Task Execution using Generic MCP Tools)

"The user can ask the agent to perform a variety of tasks that operate the computer itself via flexible generic tools (instead of less flexible, larger tools)."

The user can interact with an AI agent, which has access to a properly sandboxed MCP that is given full control of a virtual machine. The agent maps a task into multiple steps, which are determined by the model. The agent can then perform tasks using a combination of generic, hardcoded tools to operate the computer. For example, the user can ask the agent to search the Internet, write and run code, and interact with websites. Additionally, the user can configure the system to run air-gapped (without network access), and the agent will continue to complete tasks.

Feature 2 (Improved AI Knowledge via User-Supplied Documentation)

"The user can improve agent memory and knowledge by providing reliable documentation."

The user can provide reliable and relevant documents to the model to improve performance and knowledge. Providing reliable documentation allows information curated by the user to be injected to the agent, which can include internal or non-public docs. The user provided documentation is intelligently provided to the model, adding context relevant to the user's request.

Feature 3 (Versatile Deployment with Widespread Hardware Compatibility)

"The user can have up-to-date software with dynamic update of a system that has network access, or remain offline for security reasons."

The user can configure the deployment for various custom environments. Additionally, the user can select specific tools during deployment and disallow undesired tools, such as running code or Internet access. The user can utilize an installation wizard to easily download the system, which consolidates LLM, MCP, and the client locally. The user can choose an air-gapped or Internet-connected version of the system, and for Internet-facing systems, the user can receive automatic updates. For air-gapped configurations, installation is static and will not automatically update.

Novel Features/Functionalities

Some systems that can perform some computer operations currently exist, such as OpenClaw. These systems quickly expose an agent, give it tools, and allow it to work.

Our system is a complete, deployable computer-control platform that can safely operate in multiple, secure environments. A major advantage of this system is that everything is local and tuned for specific use cases, and can be tested by the Coast Guard Aux.

This architecture also enforces (not encourages) security. The AI's computer is isolated and sandboxed by default. A strict ingress/egress component is included, but does not allow the agent to interact with the host computer.

Additionally, the project supports multiple deployment types, allowing the ATA system to be supported on an enterprise-grade scale.

A unique feature of the system is agent memory as an explicit subsystem. Current systems such as OpenClaw typically only expose RAG, while our system will include a robust memory management system that allows for dynamic content storage.

Finally, the most novel/unique technical contribution is a "generic MCP." Most MCP servers are domain-specific (e.g. GitHub MCP, Browser MCP, Database MCP, etc.). Our MCP is general enough to support operating system controls, and expose the computer itself.

Algorithms and Tools

Potentially useful algorithms and software tools for each feature.

Feature 1 – Task Execution using Generic MCP Tools

- Uses a properly sandboxed MCP that is given full computer control.
    - *Python, FastMCP, Uvicorn, Linux, Docker*
- The agent maps a task into multiple steps.
    - *Gemma4 31B, Qwen3.8 27B, Transformers, PyTorch, OpenAI-Agents, Unsloth*

- The agent performs tasks using a combination of generic, hardcoded tools to operate the computer.
    - *Python, FastMCP, Linux, PyAutoGUI, Computer-control libraries/tools*
- Mapping of tasks to steps and shared tools is in the model's context window.
    - *Gemma4 31B, Qwen3.8 27B, Transformers, OpenAI-Agents, Unsloth*
- For air-gapped systems without network access, generic tools that need network access are removed via initial configuration.
    - *Python, FastMCP, Linux, Docker*

Feature 2 – Improved AI Knowledge via User-Supplied Documentation

- An MCP tool that searches the RAG for model-determined "search keywords" utilizing two types of supporting documents.
    - *Python, OpenAI-Agents, FastMCP, HuggingFace, Vector Database, text-embedding-3-large*
- A normal embedding model injects context into the prompt.
    - *text-embedding-3-large, OpenAI, HuggingFace, Python, Unsloth*

Feature 3 – Versatile Deployment with Widespread Hardware Compatibility

- Tools are selected by the user during deployment; should the user not desire a specific tool or toolset (e.g. disabling running code or Internet access), they will do so during installation wizard.
    - *Docker, FastMCP, Python, Nginx*
- Utilize A/B software for update versioning for systems connected to the Internet.
    - *Docker, GitHub*
- Systems can consolidate LLM, MCP, and client locally.
    - *Docker, Linux, FastMCP, Gemma4 31B, Qwen3.8 27B, Unsloth, HuggingFace, PyTorch*
- System can be Internet isolated (air-gapped) or connected to the Internet; dynamic installation allows for flexibility.
    - *Linux, Docker*
- For air-gapped systems without network access, this feature is not available and manual update is needed.
    - *Docker, Linux, USB (hardware)*

Technical Challenges

Main CSE-related challenges for each feature.

Feature 1 – Task Execution using Generic MCP Tools

- Uses a properly sandboxed MCP that is given full computer control.
    - *Is the MCP properly sandboxed?*
    - *Can the agent somehow leave or access something outside of the sandbox?*

    - *How do we isolate processes and resources?*
- The agent maps a task into multiple steps.
    - *Can we balance a small LLM with complex tool selection and reasoning?*
    - *What if the LLM selected can not map the task into steps?*
- The agent performs tasks using a combination of generic hardcoded tools to operate the computer.
    - *Can we find a small, generic list of tools?*
    - *What about edge-cases or obscure goals?*
- Mapping of tasks to steps and shared tools are in the model's context window.
    - *Can the small model sufficiently handle the tools in its context window?*
    - *Can we balance sufficient tools with a large enough context window?*
- For air-gapped systems without network access, generic tools that need network access are removed via initial configuration.
    - *Will the user understand how and which tools to disable?*
    - *What if the user leaves a tool enabled, and it errors out?*

Feature 2 – Improved AI Knowledge via User-Supplied Documentation

- An MCP tool that searches the RAG for model-determined "search keywords" utilizing two types of supporting documents.
    - *How do we communicate to the user that their documents must be reliable?*
    - *What if the LLM is not good at searching context with the tool?*
- A normal embedding model injects context into the prompt.
    - *How do we choose good parameters for how much, and which, context to pull from the vector store?*
    - *How do we provide this to the model efficiently?*

Feature 3 – Versatile Deployment with Widespread Hardware Compatibility

- Tools are selected by the user during deployment; should the user not desire a specific tool or toolset (e.g. disabling running code or Internet access), they will do so during installation wizard.
    - *Can we trust the user to complete this step correctly?*

For example, a tool would error if they leave Internet-requiring tools enabled on an air-gapped system.

- Utilize A/B software for update versioning for systems connected to the Internet.
    - *How can we implement the versioning system?*
    - *How can we make this step automatic once we push an update?*
- Systems can consolidate LLM, MCP, and client locally.
    - *How can we integrate all 3 of these systems on resource constrained systems?*
    - *What minimum system requirements are needed?*
- System can be Internet isolated (air-gapped) or connected to the Internet; dynamic installation allows for flexibility.
    - *How do we ensure our system continues to function without Internet?*

    - *What happens if it tries to access the Internet and is denied?*
- For air-gapped systems without network access, this feature is not available and manual update is needed.
    - *Would we need to dynamically test for Internet access?*
    - *How will users manually update?*

Milestones

The first three project milestones are present below. These reference the System Requirements specified in the ConOps https://ai-task-agent.atlassian.net/wiki/spaces/ATA/pages/6127617 .

Feature statuses are as follows:

1. **Proof-of-Concept (POC)**: the lowest completion status, showing that a feature or capability is possible. Is not finished, may be bug-heavy, and may not meet any requirements.
2. **Prototype**: a POC that meets some requirements and is fairly reliable.
3. **Minimum Viable Product (MVP)**: stage at which the subsystem meets the minimum requirements and bug/security testing to be considered complete.
4. **Production (Prod)**: established when the MVP has undergone thorough security and bug testing and meets or exceeds all requirements. O&M may occur in this stage.

Milestone 1 Itemized Tasks (Sept. 28):

Accomplishing this milestone means that the project is fully planned and ready for development.

- Compare and select technical tools for Large Language Models (LLMs) (req. 1), Deployment Environment (2), Model Context Protocol Server (3), Linux (3.4, 4), Browser (3.10.1), Email (3.10.3), PDF reading (3.11.1), Communications (3.12.1-3.12.3), Retrieval-Augmented Generation (RAG) (5.1), Agentic Dynamic Memory (5.3), Context Window Management (5.5), Fast Context (5.6), and Graphical User Interfaces (6.1).
- Provide small ("hello world") demos to evaluate the tools for Large Language Models (LLMs) (req. 1), Deployment Environment (2), Model Context Protocol Server (3), Linux (3.4, 4), Browser (3.10.1), Email (3.10.3), PDF reading (3.11.1), Communications (3.12.1-3.12.3), Retrieval-Augmented Generation (RAG) (5.1), Agentic Dynamic Memory (5.3), Context Window Management (5.5), Fast Context (5.6), and Graphical User Interfaces (6.1).
- Resolve technical challenges: Task Execution using Generic MCP Tools, Improved AI Knowledge via User-Supplied Documentation, Versatile Deployment with Widespread Hardware Compatibility. (see Technical Challenges section for full list)

- Compare and select collaboration tools for software development, documents/presentations, communication, task calendar
- Create Requirement Document
- Create Design Document

- Create Test Plan

Milestone 2 Itemized Tasks (Oct. 26):

Accomplishing this milestone means that the project has done initial development and a working system is delivered.

- Implement, test, and demo **Prototype for Task Execution using Generic MCP Tools**, without advanced security features implemented (feature 1).
- Implement, test, and demo **Prototype AI Agent**, with full connection to Prototype Task Execution using Generic MCP Tools (feature 1).
- Implement, test, and demo **POC RAG memory**, with basic documentation tested and linked to Prototype AI Agent (feature 2).
- Implement, test, and demo **Prototype for Versatile Deployment with Widespread Hardware Compatibility**, with the system functional to current development on a new install on a new system (feature 3).

Milestone 3 Itemized Tasks (Nov. 23):

Accomplishing this milestone means that most features are MVP status and that a fully function system is complete, albeit not fully production-ready.

- Implement, test, and demo **MVP for Task Execution using Generic MCP Tools**, without advanced security features implemented (feature 1).
- Implement, test, and demo **MVP AI Agent**, with full connection to Prototype Task Execution using Generic MCP Tools (feature 1).
- Implement, test, and demo **Prototype RAG memory**, with basic documentation tested and linked to Prototype AI Agent (feature 2).
- Implement, test, and demo **MVP for Versatile Deployment with Widespread Hardware Compatibility**, with the system functional to current development on a new install on a new system (feature 3).

Task Matrix for Milestone 1

Task	Jadon	Jacob E	Ioannis	Jacob L
Compare and Select Technical Tools	LLMs, Deployment Environment, RAG	Browser, Email, Agentic Dynamic Memory, GUI	MCP Server, PDF Reading, Context Window Management	Linux, Communications, Fast Context
“Hello World” Demos	LLMs, Deployment Environment, RAG	Browser, Email, Agentic Dynamic Memory, GUI	MCP Server, PDF Reading, Context Window Management	Linux, Communications, Fast Context

Resolve Technical Challenges	Is the MCP properly sandboxed? Can the agent somehow leave or access something outside of the sandbox? How do we isolate processes and resources? Can we balance a small LLM with complex tool selection and reasoning? What if the LLM selected cannot map the task into steps? How do we choose good parameters for how much, and which, context to pull from the vector store? How do we provide this to the model efficiently?	How do we communicate to the user that their documents must be reliable? What if the LLM is not good at searching context with the tool?	Can we find a small, generic list of tools? What about edge-cases or obscure goals? Can the small model sufficiently handle the tools in its context window? Can we balance sufficient tools with a large enough context window? Will the user understand how and which tools to disable? What if the user leaves a tool enabled, and it errors out? How can we integrate all 3 of these systems on resource constrained systems? What minimum system requirements are needed?	Can we trust the user to complete the installation step correctly? For example, a tool would error if they leave Internet-requiring tools enabled on an air-gapped system. How do we ensure our system continues to function without Internet? What happens if it tries to access the Internet and is denied?

	How can we implement the versioning system? How can we make this step automatic once we push an update? Would we need to dynamically test for Internet access? How will users manually update?			
Compare and Select Collaboration Tools	Task Calendar	Documents/Presentations	Communication	Software Development
Requirement Document	Write 40%	Write 20%	Write 20%	Write 20%
Design Document	Write 30%	Write 10%	Write 20%	Write 40%
Test Plan	Write 10%	Write 40%	Write 30%	Write 20%

Approval from Faculty Advisor

- "I have discussed with the team and approve this project plan. I will evaluate the progress and assign a grade for each of the three milestones."
- Signature: _____ Date: _____