

Project P4: Agentic Computer Interaction with Vision and Speech

New Attempt

- Due Dec 10, 2025 by 11:59pm
- Points 100
- Submitting a website url or a file upload

Overview

This is part 4 of our semester-long project: Intelligent Human Assistant. P4 brings together the concepts from previous assignments to create an agentic system that interacts with your computer through vision and speech.

The system enables voice-controlled screen interaction by capturing screenshots, running OCR to detect UI elements and their positions, mapping numbered labels to UI elements, and executing mouse clicks on target elements based on voice commands.

Core capability: "Click the Settings button" → System captures screen → OCR identifies numbered labels → System calculates coordinates → Executes click

This assignment uses a hybrid architecture combining **mcp-agent** for workflow orchestration with guaranteed sequential execution and **FastMCP** for streamable HTTP transport.

Learning Objectives

By completing this assignment, you will:

1. Build an agentic system that perceives and acts on computer interfaces
2. Use OCR (PaddleOCR) for UI element detection and localization
3. Implement MCP workflows with **guaranteed sequential** execution
4. Handle coordinate mapping from OCR detection to actual screen coordinates
5. Understand the perception-action loop in real-time desktop automation

System Architecture

- FastMCP Server:
 - This is **NOT new**. This handles the communication layer for the MCP protocol over HTTP. We are all familiar with this.
- mcp-agent Workflows:

- **This is NEW!** Workflows are a methodical way to execute multiple functions in agentic interaction, where the execution order of each function matters. Each workflow consists of multiple tasks/functions decorated with @mcp_agent_app.workflow_task that execute in order. Each MCP tool can call one or more Workflows. Please familiarize yourself with the documentation: <https://github.com/lastmile-ai/mcp-agent> ↗ (<https://github.com/lastmile-ai/mcp-agent>)
- Open WebUI:
 - **This is NEW.** This is an open-source, industry-standard user interface for chatbots. We discussed how to set it up in prior classes. It has built-in support for the MCP architecture and tool calling. It replaces our old MCP client or Streamlit-based UI.

What We Provide

We provide the following scaffolding code:

[.env](https://psu.instructure.com/courses/2416221/files/185391562?wrap=1) (<https://psu.instructure.com/courses/2416221/files/185391562?wrap=1>) ↴
[\(\[https://psu.instructure.com/courses/2416221/files/185391562/download?download_frd=1\]\(https://psu.instructure.com/courses/2416221/files/185391562/download?download_frd=1\)\)](https://psu.instructure.com/courses/2416221/files/185391562/download?download_frd=1)

This is a template for your environment settings. Place it in the same directory as the server code and update the key values to match your setup.

[p4_mcp_server_starter.py](https://psu.instructure.com/courses/2416221/files/185391541?wrap=1) (<https://psu.instructure.com/courses/2416221/files/185391541?wrap=1>) ↴
[\(\[https://psu.instructure.com/courses/2416221/files/185391541/download?download_frd=1\]\(https://psu.instructure.com/courses/2416221/files/185391541/download?download_frd=1\)\)](https://psu.instructure.com/courses/2416221/files/185391541/download?download_frd=1)

This file contains a minimal viable product. It includes:

- **VanillaWorkflow and vanilla_workflow_tool:** A basic workflow demonstrating the **mcp-agent** pattern. This workflow reads the .env file to get the screenshot path, captures a screenshot, and announces completion via text-to-speech. Use this as a reference for implementing your workflows.
- **Several utility functions:**
 - speak_text(text) - Google TTS function for audio feedback (this is a thread-safe audio playback method)
 - load_env_file() - Loads environment variables from .env file
 - get_ocr() - Singleton PaddleOCR instance
 - extract_ascii_digits(text) - Extracts only ASCII digits from OCR text
 - Screen dimension detection and scaling factors (scale_x, scale_y)

[README.md](https://psu.instructure.com/courses/2416221/files/185391545?wrap=1) (<https://psu.instructure.com/courses/2416221/files/185391545?wrap=1>) ↴
[\(\[https://psu.instructure.com/courses/2416221/files/185391545/download?download_frd=1\]\(https://psu.instructure.com/courses/2416221/files/185391545/download?download_frd=1\)\)](https://psu.instructure.com/courses/2416221/files/185391545/download?download_frd=1)

This file contains detailed instructions, including a system prompt that instructs the LLM how to use your MCP tools. This configures the P4 assistant in Open WebUI.

What You Implement

You will implement **two workflows** and their corresponding MCP tools:

1. CaptureScreenWithNumbers Workflow

This workflow captures the screen and creates a mapping between numbered labels and UI element text. The workflow should:

- Announce "start listening" via TTS and wait for the UI to display numbered labels (you need to enable the Voice Control app in Mac or Voice Access app in Windows)
- Announce "show numbers" to trigger numbered label display on screen
 - **Note that you may need to issue other speech commands (e.g., bring an app to the foreground) to get a clean screenshot**
 - **You may also consider reducing the screen resolution so that a screenshot takes less memory, and the OCR/VLM can process faster**
- Capture a screenshot using PIL.ImageGrab
- Run PaddleOCR (or any better OCR or VLM model) on the screenshot to detect all text elements
- Create mappings between numbered labels and UI text (handle both embedded numbers like "1Settings" and adjacent number-text pairs)
- Save mappings to a JSON file with structure: {"mappings": [{"number": "1", "text": "Settings", "center_x": 100, "center_y": 200}, ...]}
- Announce "stop listening" when complete

Tool: capture_screen_with_numbers_tool() - Exposes the workflow as an MCP tool.

2. ClickWorkflow Workflow

This workflow finds and clicks on a target UI element. The workflow should:

- Read the OCR metadata of the most recently captured screenshot from the JSON file created by CaptureScreenWithNumbers
- Search for the target element by text matching (case-insensitive substring)
- Convert OCR coordinates to screen coordinates using the scaling factors
- Move the mouse to the target position using pyautogui.moveTo()
- Execute the click using pyautogui.click()
- Announce "clicked on [target]" via TTS
- **Tool: click_workflow_tool(target: str)** - Takes a target name (e.g., "Settings", "File") and executes the click workflow.

Expected Interaction Flow

Here's how a typical interaction proceeds:

This video may display YouTube ads.

[Continue to YouTube content](#)

Submission Guidelines

Submit a zip file containing:

- **Demo Video:** Your demo must contain **at least 3 different screens (each showing different apps)**. Demonstrate both successes and failures. Show the full workflow from voice command to executed action.
- **Source Code:** All Python files, including your implementations of CaptureScreenWithNumbers, ClickWorkFlow, and the corresponding tools
- **Report:** Explain your approach, challenges faced, and how you handled edge cases like OCR errors or coordinate mismatches.