International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Sluely: A Review of the Always-on AI Assistant That Never Leaves Your Machine

Shubham Chavan, Mihir Patil, Sujal Bhalerao, Kunal Yendole
Department of Artificial Intelligence and Machine Learning, Zeal Polytechnic, Pune, Maharashtra, India Academic Year 2026–2027 | Project Guide: Maheshwari Bhalshankar

Abstract— Always-on AI assistants are a growing category of desktop software that interpret a user's active digital context and provide help without repeated manual prompts. This paper reviews **Sluely**, a local-first desktop AI assistant designed to remain available in the background while keeping its core context-processing and inference workflow on the user's machine. The reviewed design combines periodic screen sampling, optical character recognition (OCR) and change detection, system and microphone audio capture, local speech-to-text, a rolling context buffer, and locally hosted language-model inference through Ollama. Results are presented through a lightweight overlay intended to support in-context questions, meeting notes, and action-item extraction. This paper examines the workflow, system modules, privacy model, applications, and engineering limitations. The current project description identifies screen sampling, on-device audio capture, context buffering, Ollama streaming, and the overlay as implemented capabilities; task-mode switching, multi-day episodic memory, and broader proactive triggers remain planned extensions. This is a design-oriented review and does not claim benchmark performance or independently verified accuracy.

Keywords— Always-on AI assistant, local-first AI, desktop assistant, Ollama, speech-to-text, screen context, OCR, privacy-preserving AI.

I. INTRODUCTION

Desktop work is distributed across browsers, documents, code editors, meetings, messaging tools, and operating-system applications. Users frequently switch between these tools to retrieve information, summarize discussions, or understand material currently visible on screen. Conventional chat-based assistants depend on users selecting context and writing a prompt. Although useful, this interaction model can interrupt work and omit information that was visible or spoken moments earlier.

Always-on assistants explore a different pattern: they remain available in the background, observe selected forms of user-authorized context, and provide help when requested or when a relevant trigger is detected. This creates an engineering trade-off. More context may improve relevance, but continuous screen and audio access increases privacy, consent, security, and resource-use concerns.

Sluely is designed around this trade-off. It is a local-first desktop assistant that combines screen and audio context with a locally hosted language model. Its stated design keeps screen samples, OCR output, audio processing, transcription, context buffers, and model inference on the local device. Local processing can reduce routine cloud exposure, but does not itself guarantee privacy: captured information remains sensitive and may include information about other people.

II. PROBLEM STATEMENT AND OBJECTIVES

Conventional desktop assistance often requires users to switch applications, copy information, describe their current task, or manually upload screenshots and recordings. This repeated context assembly interrupts workflow and may omit relevant details. An assistant that remains available can reduce this friction, but must address the risks of continuous context collection, unnecessary retention, and inaccurate generated responses.

The problem addressed by Sluely is how to provide timely, context-aware desktop assistance while keeping the core processing workflow local and maintaining clear user control over screen and audio capture.

A. Objectives

The objectives of Sluely are to: (1) maintain an always-available desktop assistant; (2) collect selected screen and audio context with user permission; (3) extract readable information from screen and speech inputs; (4) organize recent events in a rolling context buffer; (5) use a locally hosted language model to generate contextual responses; (6) stream answers through a lightweight overlay; and (7) support tasks such as answering on-screen questions, meeting-note preparation, and action-item extraction.

B. Scope

This review focuses on the product concept and capabilities reported as live in the project description. It does not assume every proposed feature is complete, nor does it report controlled user studies, latency measurements, transcription accuracy, energy consumption, or model-quality benchmarks. Task-mode switching, multi-day episodic memory, and broader proactive triggers are treated as planned extensions.

III. EXISTING AND PROPOSED SYSTEM

A. Existing System

Typical desktop AI tools rely on explicit prompts, manual file uploads, browser extensions, or meeting bots. Users must often select and transfer context before receiving assistance. Cloud-based assistants may transmit prompts and uploaded information to remote services. These approaches can be useful, but repeated context preparation may interrupt work and can create privacy concerns when sensitive material is involved.

B. Proposed System

Sluely proposes a connected local workflow: capture selected screen and audio context, convert relevant content into text, organize it in a rolling buffer, and use a locally hosted Ollama model to generate a response. The response is streamed to a lightweight desktop overlay. The design aims to reduce manual context switching while keeping core inference on the user's machine.

C. Expected Benefits

The intended benefits are reduced prompt preparation, more continuity between desktop tasks, support for meeting notes and action items, and reduced routine dependence on remote inference. These are design goals rather than measured outcomes. Actual usefulness depends on capture quality, context selection, model capability, and user control.

IV. METHODOLOGY

A. Project Selection and Local Scanning

The application starts with the desktop session and the user-selected capture permissions. It initializes the screen and audio components, checks local model availability, and prepares the context buffer. Capture is limited by the permissions and settings selected by the user.

The proposed methodology is organized into a sequence of local processing stages, beginning with context collection and continuing through extraction, inference, and output presentation.

B. Screen and Audio Context Collection

The desktop application collects selected screen frames and authorized audio input. Screen sampling provides visual context from the active workspace, while audio capture can include system sound and microphone input. The capture process should be permission-based and visibly indicated to the user.

B. Text and Speech Extraction

Screen frames are processed to identify changes and extract visible text through OCR. Audio is passed to a local speech-to-text component to produce text segments. Both sources are timestamped or otherwise ordered so that later processing can associate events with the correct context.

C. Context Buffer Construction

Extracted screen text, detected changes, and transcription segments are organized into a rolling context buffer. The buffer is intended to retain recent, relevant information rather than an unlimited record of activity. Retention limits and clearing controls are important to reduce unnecessary data storage.

D. Local AI Response Generation

When the user asks a question or an enabled trigger is activated, the application selects relevant buffer entries and sends them to a locally served language model through Ollama. The model response is streamed back to the desktop application. Response quality depends on the selected model, prompt construction, and accuracy of the collected context.

E. Overlay Presentation

The overlay displays the generated response without requiring the user to leave the active application. It may present explanations, meeting-note drafts, or action items. Generated content should remain reviewable and editable by the user.

V. SYSTEM WORKFLOW

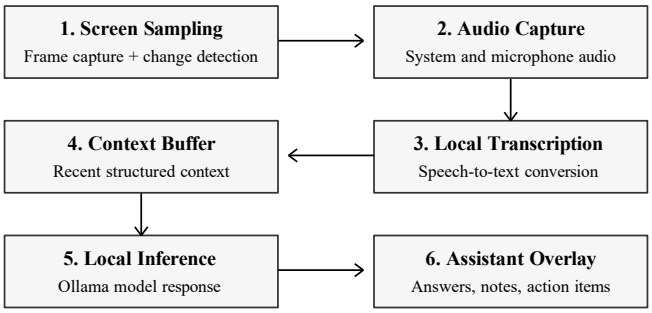


Fig. 1. Overall workflow of the proposed Sluely platform.

The workflow begins with screen and audio context collection. Extracted text and transcription are organized into a rolling buffer. Relevant context is passed to local inference, and the streamed response is displayed through the assistant overlay.

VI. SYSTEM MODULES

A. Screen Sampling and Change Detection Module

This module captures selected screen frames and identifies meaningful changes. Change detection can limit redundant processing, while OCR converts visible text into structured input. Sampling frequency and region selection influence responsiveness and system resource use.

B. Audio Capture and Transcription Module

This module captures permitted system and microphone audio and converts speech into text using a local transcription component. It supports meeting-oriented assistance without requiring a separate bot to join a call. Participant consent and applicable recording requirements still apply.

C. Context Management Module

This module organizes recent screen-derived text, detected changes, and transcription segments into a rolling context buffer. It should remove stale material, preserve relevant ordering, and apply clear retention limits.

D. Local Inference Module

The inference module connects to an available Ollama model, constructs a prompt from the user's question and relevant context, and streams the generated response. Model size and hardware affect response quality, latency, and memory use.

E. Overlay and Assistance Module

The overlay presents responses, summaries, and suggested action items. It should remain unobtrusive, avoid stealing focus, and provide controls to dismiss or pause assistance.

SLUELY MODULES AND TECHNOLOGIES

Module	Technology	Output
Screen and audio capture	OS APIs; capture pipeline	Screen frames and audio
Text extraction	OCR; speech-to-text	Readable context
Context management	Rolling context buffer	Recent context records
AI inference	Local Ollama model	Streaming responses
Overlay	Desktop UI	Answers and notes

VII. TOOLS AND TECHNOLOGIES

Screen capture and OCR: used to obtain visual context and extract readable text. **Audio and speech-to-text:** used to convert authorized speech input into text. **Ollama:** provides a local model-serving interface and streaming responses [1]. **Desktop overlay:** presents the assistant's output over the active workspace. The exact OCR, transcription, and desktop UI libraries depend on the implementation version.

VIII. PRIVACY AND SECURITY

Local-first processing aims to keep screen samples, OCR text, audio capture, transcription, context buffering, and inference on the device. This reduces routine transmission of raw context to a remote inference provider, but is not an unconditional privacy guarantee. Local files, logs, prompts, and temporary buffers can still expose sensitive information if retained or accessed improperly.

A responsible implementation should make capture state visible, provide pause and stop controls, minimize retained data, secure local storage, and disclose whether diagnostics or telemetry leave the device. Users must consider the rights and expectations of people whose voices or information may be captured.

IX. CURRENT CAPABILITIES AND DEVELOPMENT STATUS

The current project description identifies screen sampling, on-device audio capture, a rolling context buffer, Ollama response streaming, and the lightweight overlay as the live core loop. These capabilities form the basis of the system reviewed in this paper.

IX. EVALUATION PLAN AND LIMITATIONS

A. Evaluation Measures

A meaningful evaluation should use representative desktop tasks and consented audio samples. Suggested measures include: (1) screen-context extraction precision and recall; (2) transcription word error rate across languages and noise conditions; (3) answer relevance and factual correctness; (4) end-to-end response latency; (5) CPU, GPU, RAM, and energy use during continuous operation; (6) overlay interruption rate and user-perceived distraction; and (7) data-retention and network-traffic audits. Results should identify device configuration, model version, test set, and measurement procedure.

B. Technical Limitations

Local models may be less capable than larger hosted systems and can produce incorrect or incomplete answers. Continuous screen sampling and audio processing can consume significant resources. OCR may misread content, speech recognition may struggle with accents or overlapping speakers, and the context buffer may omit important details or include irrelevant material. Hardware requirements may limit use on lower-end computers.

C. Privacy and Operational Limitations

Local processing does not eliminate risks from unauthorized access, insecure storage, accidental capture, or insufficient consent. Meeting summaries may misattribute speakers or omit qualifiers. Proactive suggestions can become intrusive if triggers are too sensitive. User control, transparent indicators, conservative defaults, and easy deletion are essential.

D. Review Limitations

This paper is based on the described product architecture and documented enabling technologies. It does not independently inspect the complete source code or validate every platform-specific implementation. No experimental results, benchmark values, or comparative performance claims are presented because a controlled evaluation dataset and measurement report were not supplied.

X. CONCLUSION

Sluely represents a local-first approach to always-available desktop assistance. Its central idea is to combine recent screen and audio context with local language-model inference and present help through a lightweight overlay. The described live workflow—screen sampling, on-device audio capture, a rolling context buffer, Ollama streaming, and overlay presentation—offers a foundation for in-context questions and meeting-oriented assistance. Its value depends not only on model quality but also on reliable context selection, low-latency processing, transparent capture controls, and responsible handling of sensitive information.

Future work should prioritize task-mode switching, carefully bounded episodic memory, configurable proactive triggers, cross-platform permission handling, and systematic evaluation. Before making strong claims about accuracy, privacy, or efficiency, the project should publish reproducible measurements and a clear data-flow description covering local storage, network activity, retention, and deletion.

REFERENCES

[1] Ollama, “Ollama API Documentation.” Available: https://docs.ollama.com/api

[2] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust Speech Recognition via Large-Scale Weak Supervision,” Proceedings of the 40th International Conference on Machine Learning, PMLR 202, pp. 28492–28518, 2023. Available: https://proceedings.mlr.press/v202/radford23a.html

[3] Apple Support, “Control access to screen and system audio recording on Mac.” Available: https://support.apple.com/guide/mac-help/control-access-screen-system-audio-recording-mchld6aa7d23/mac

[4] Microsoft Support, “Windows camera, microphone, and privacy.” Available: https://support.microsoft.com/en-us/windows/privacy/windows-camera-microphone-and-privacy