International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 01 | Jan 2026 www.irjet.net p-ISSN: 2395-0072

Smart Macro Tool For Automation of Repetitive Digital Tasks

Ayaan Irfan Surve [1], Mohammed Hayat Azad[2], Jatin Patil[3],Toshik Gawade[4], Hafsah Siddique[5]

[1]Ayaan Irfan Surve, Backend Developer
[2]Mohammed Hayat Azad, Full Stack Devloper (Team Leader)
[3]Jatin Patil, Frontend Developer
[4]Toshik Gawade, Frontend Developer
[5]Hafsah Siddique, Professor, Dept. of Computer Engineering, Abdul Razzak Kalsekar Polytechnic , Maharastra, India (Team Mentor)

Abstract - *In the modern digital workspace, knowledge workers are frequently bottlenecked by repetitive tasks such as formatting slides, restructuring file directories, and transferring data between incompatible document formats. While cloud-based generative AI offers potential solutions, privacy concerns and latency often hinder its adoption for handling sensitive local files. This paper presents the design and implementation of a Smart Macro Tool, a desktop-native automation agent designed to eliminate digital drudgery. The system features a modular architecture comprising a Generative PowerPoint Engine capable of creating styled presentations from natural language prompts, a Smart PDF Writer, a Directory Constructor for instant file system organization, and a Neural Action Recorder leveraging pynput for reproducible workflow automation. By utilizing a "Sidecar" architecture with a lightweight frontend and a local Python backend, the tool ensures all processing—including AI inference—occurs locally, guaranteeing data sovereignty. We demonstrate that this approach significantly reduces time-on-task for common administrative workflows without compromising user privacy*

Key Words: **Smart Macro Tool, Desktop Automation, Privacy-First Architecture, Local LLMs, Tauri Framework, Python Sidecar, Generative PowerPoint, Smart PDF Writer, Action Recorder, Directory Maker, pynput, python-pptx, Human-in-the-Loop, Offline AI, Program Synthesis, Workflow Optimization**

1. INTRODUCTION

1.1 The Problem Of Digital Repetition

Despite significant advancements in software usability and interface design, a substantial portion of everyday computer interaction continues to be dominated by repetitive, low-value operational tasks. Users routinely engage in manual activities such as creating standardized directory structures for new projects, transferring and formatting text into presentation slides, or executing identical sequences of clicks to generate and export reports. While individually trivial, these recurring “micro-tasks” accumulate over time, fragmenting user attention and disrupting cognitive flow. As a result, mental resources that could otherwise be allocated to analytical reasoning, creative problem-solving, or strategic decision-making are instead consumed by procedural overhead.[1],[2]

This persistent reliance on manual execution highlights a gap between modern software capabilities and actual user workflows. Although many applications offer automation features, they are often inaccessible or impractical for non-technical users. Existing solutions tend to fall into two opposing extremes. On one end are rigid automation scripts and macros that require programming expertise, detailed configuration, and ongoing maintenance—creating a steep learning curve and limiting adoption beyond technically proficient users. On the other end are cloud-based artificial intelligence tools that abstract complexity but necessitate the transfer of potentially sensitive data to external servers. Such data exfiltration introduces privacy, compliance, and security concerns, particularly within enterprise and regulated environments where data sovereignty and confidentiality are critical.[1],[3],[13]

Consequently, there remains an unmet need for automation approaches that are both user-friendly and secure—solutions that can seamlessly integrate into local workflows, reduce repetitive effort, and enhance productivity without demanding specialized technical knowledge or compromising organizational data governance policies.

1.2 Proposed Solution

We propose a Smart Macro Tool, a desktop-based application designed to function as an intelligent intermediary layer between the user and the underlying operating system. Unlike traditional macro systems—which are typically confined to individual applications, such as Excel VBA or application-specific scripting engines—this solution operates at the system level, enabling automation that spans multiple software environments and workflow contexts. By abstracting low-level interactions into higher-level intent-driven actions, the tool aims to significantly reduce repetitive effort while remaining accessible to non-technical users.[1],[3],[12]

The proposed system incorporates several key capabilities. First, it enables generative content creation, automating the production of structured artifacts such as PowerPoint presentations and PDF documents. This is achieved through the use of locally deployed Large Language Models (LLMs), which analyze user-provided inputs to determine appropriate tone, organization, and formatting. By executing all inference locally, the system preserves data privacy and avoids reliance on external cloud services, making it suitable for security-sensitive and enterprise environments.

Second, the tool supports cross-application macro recording and execution. Leveraging low-level input hooks at the operating system level, it can capture and replay complex user interactions—including keyboard input, mouse movements, and application switching—across heterogeneous software systems. This approach overcomes the limitations of application-scoped macros and enables end-to-end automation of real-world workflows that typically span multiple tools.

Finally, the system provides structural automation capabilities, allowing users to define abstract, text-based representations of repetitive structures—such as directory hierarchies or standardized file layouts—which are then automatically translated into concrete file system objects. This feature reduces setup time for new projects and enforces consistency across organizational workflows.

Collectively, these capabilities position the Smart Macro Tool as a flexible, secure, and extensible automation platform that bridges the gap between rigid scripting solutions and opaque cloud-based AI services.

2. SYSTEM ARCHITECTURE

The Smart Macro Tool is implemented as a desktop-resident agent in order to ensure direct, low-latency access to local system resources, including the file system, input devices, and application processes. Operating locally allows the tool to observe and automate user interactions at the operating system level while maintaining strict control over data locality and execution context.[14]

2.1 Hybrid "Sidecar" Design

To achieve an effective balance between runtime performance, security, and development agility, the system adopts a hybrid sidecar architecture that cleanly separates the user interface layer from the computational core.

The frontend is developed using Tauri, which provides a lightweight and secure desktop interface rendered through a native web-view. Compared to Electron-based frameworks, Tauri significantly reduces memory footprint and startup overhead by avoiding the bundling of a full browser runtime. This design choice ensures a responsive user experience even on resource-constrained systems while preserving the flexibility of modern web technologies for interface development.

The backend, referred to as the sidecar, is implemented as a standalone Python executable responsible for handling computationally intensive operations. These include file system manipulation, macro recording and replay logic, and local AI inference. By isolating heavy workloads within the sidecar process, the architecture prevents long-running tasks—such as generating large presentation decks or processing extensive automation sequences—from blocking or degrading the responsiveness of the user interface. Inter-process communication between the frontend and backend enables efficient coordination while maintaining strong separation of concerns.[8],[14]

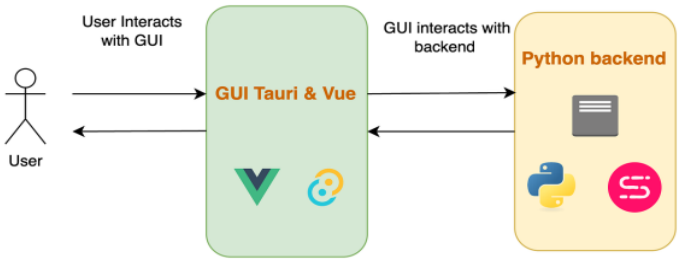

Fig 1- System Architecture

2.2 Local Intelligence Engine

User privacy and data sovereignty are treated as foundational design requirements. To this end, the system integrates a local intelligence engine based on quantized, on-device Large Language Models (LLMs), such as Llama-3 or Mistral. These models are executed locally using efficient inference runtimes including Ollama or llama.cpp, enabling natural language understanding and content transformation without reliance on external cloud services.

This approach allows the tool to interpret and act upon high-level natural language instructions—such as "make this sound professional" or "summarize this into slides"—while ensuring that all user data remains confined to the local machine. By eliminating network dependency for language processing, the system mitigates security risks, supports offline operation, and complies with enterprise privacy and regulatory constraints, making it suitable for deployment in sensitive or restricted computing environments.[3],[12]

3. CORE MODULES AND METHODOLOGY

3.1 Generative PowerPoint Maker

The Generative PowerPoint Maker module is designed to automate the transformation of high-level ideas into well-structured, visually consistent presentation decks. By abstracting presentation design into intent-driven inputs,

the module significantly reduces the manual effort typically required to create professional slides.

At the core of this module is a prompt-to-slide generation pipeline. Users begin by specifying a topic along with a desired communicative tone, such as casual, professional, or academic. Based on these inputs, the locally deployed LLM generates a logical slide outline and corresponding textual content that adheres to the selected tone and presentation context. This ensures that both the structure and language of the presentation are aligned with user intent.

For implementation, the system employs the python-pptx library to directly manipulate the OpenXML structure of .pptx files. This approach enables fine-grained control over slide layouts, placeholders, typography, and formatting elements without relying on automating the PowerPoint graphical user interface. By bypassing UI-level automation—often fragile and error-prone—the module achieves greater reliability, repeatability, and cross-version compatibility.[7],[2]

To further refine output quality, the system supports tone adaptation through contextual prompt engineering. Prior to content generation, predefined system prompts are injected into the LLM's context window, such as instructing the model to adopt the voice of a corporate consultant or an academic presenter. This mechanism ensures stylistic consistency across slides while allowing flexibility in communicative intent.

3.2 Action Recorder

Certain automation scenarios involve graphical user interface interactions that cannot be effectively addressed through file system APIs or application-level scripting. For such cases, the tool includes an Action Recorder, implemented as a "watch-and-repeat" automation module.[9],[4]

The recorder operates through low-level input hooking, leveraging the pynput library to monitor keyboard and mouse events at the operating system level. Rather than relying on static screen coordinates, the system captures temporal event streams, enabling it to reproduce complex interaction sequences involving application switching, conditional waits, and non-linear workflows.

Captured interactions are stored through macro serialization into a compact, human-readable JSON format. This representation allows users to persist and manage reusable workflows—such as logging into enterprise systems, navigating dashboards, or exporting reports—and replay them on demand with minimal effort.

To ensure operational safety and prevent unintended behavior, the action replay engine executes within a separate thread and incorporates built-in safety mechanisms, including a global hotkey-based kill switch. This design allows users to immediately terminate macro execution in the event of unexpected behavior, thereby reducing the risk of uncontrolled automation.[1]

3.3 Smart Directory Maker

The Smart Directory Maker module addresses the frequent need for rapidly scaffolding standardized project structures, a task commonly encountered by developers, researchers, and project managers.

The system accepts a textual representation of directory hierarchies, such as indented lists or ASCII-style tree formats, allowing users to describe complex folder layouts in a concise and intuitive manner. This representation is processed through a recursive parsing algorithm that interprets indentation depth—based on tabs or spaces—as hierarchical relationships within the directory tree.

Once parsed, the system traverses the resulting virtual structure and programmatically creates the corresponding folders and files on disk using native file system operations (e.g., os.mkdir and file creation calls). For example, a simple textual input describing a project root with source and documentation subdirectories is automatically translated into a fully instantiated directory hierarchy, including placeholder files where specified. This capability dramatically reduces setup time while enforcing consistency across projects.[2]

3.4 Smart PDF Writer

The Smart PDF Writer module focuses on the intelligent generation and restructuring of document content into publication-ready PDF formats. It is designed to handle both newly generated content and existing unstructured text.

At its core, the module incorporates a dynamic layout engine built on mature PDF generation libraries such as Report Lab or Weasy Print. These tools enable the conversion of raw text or Markdown inputs into visually polished documents with precise control over page layout, typography, and spacing.

In addition, the system applies semantic-aware formatting by analyzing input text to identify structural elements such as headings, lists, tables, and emphasis. Based on this analysis, the module automatically applies appropriate typographic rules, including consistent margins, font hierarchies, and spacing conventions. This eliminates the need for manual layout adjustments while ensuring that generated PDFs meet professional documentation standards.[10],[11]

4. IMAGE RECOGNITION AND OCR

To improve the robustness and reliability of the Action Recorder, the system integrates basic computer vision capabilities that enable automation to adapt to dynamic and visually changing user interfaces.

Traditional input-replay mechanisms often rely on fixed screen coordinates, which can fail when interface layouts change due to window resizing, resolution differences, or application updates. To address this limitation, the proposed tool incorporates UI element detection techniques that allow recorded macros to remain functional even when target elements shift position.

During the recording phase, the system can capture a visual snapshot of the target interface element, such as a button, icon, or text label. This visual reference is stored alongside the macro metadata. During macro replay, instead of clicking at predetermined coordinates, the system analyzes the current screen state to locate the element's updated position.

Element localization is achieved through template matching techniques using Open CV, enabling the system to identify visually similar regions within the screen image. For interfaces where textual labels are more stable than visual styling, the system may alternatively employ optical character recognition (OCR) via Tesseract to detect and locate text-based UI elements. Once the element is identified, its current coordinates are computed and used to execute the intended interaction.

By incorporating computer vision–based element recognition into the automation pipeline, the Action Recorder significantly enhances resilience against interface variability. This approach reduces macro fragility, minimizes maintenance overhead, and enables reliable automation across a wider range of applications and display configurations.[6]

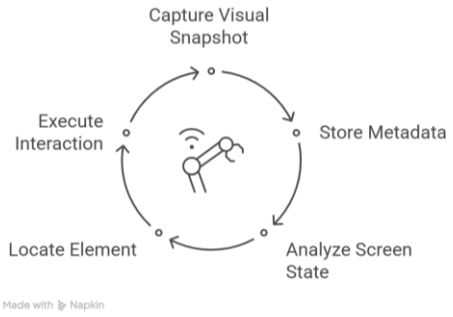

Fig 2: Image Recognition Flow in Action Recorder

5. PRIVACY AND SECURITY ANALYSIS

Given the tool's ability to access the local file system and record low-level user inputs, security and trustworthiness are treated as critical design priorities. The system is explicitly engineered to minimize risk exposure while maintaining transparency and user control.

First, the tool enforces a local-only execution model. All file-related operations—including the generation and modification of PowerPoint presentations, PDF documents, and directory structures—are performed entirely on the user's machine. At no point is file content transmitted to external servers or third-party services. This design eliminates risks associated with data leakage, unauthorized access, and compliance violations, making the tool suitable for enterprise and regulated environments.

Second, the backend Python sidecar operates within a sandboxed execution environment. Its permissions are deliberately constrained to restrict file system access exclusively to directories that have been explicitly authorized by the user. This containment strategy reduces the attack surface and prevents accidental or malicious modification of sensitive system areas. By enforcing scoped access controls, the system aligns with the principle of least privilege.

Finally, the tool emphasizes operational transparency, particularly within the Action Recorder module. Before any macro is executed, users are presented with a clear, human-readable log of the recorded steps and intended actions. This pre-execution review mechanism allows users to inspect, validate, and understand automation behavior in advance; mitigating the "black box" effect commonly associated with intelligent automation agents. As a result, users retain full awareness and control over automated workflows, fostering trust and safe adoption.[3],[1]

6. FUTURE SCOPE

The current implementation establishes a strong foundation for the evolution of a comprehensive, intelligent personal desktop assistant. Building upon this base, future development efforts will focus on enhancing both agency—the system's ability to independently plan and execute complex workflows—and intelligence—its capacity to understand context, intent, and visual semantics. Three primary directions have been identified for continued research and development.

6.1 Autonomous Multi-Agent Orchestration

At present, the system executes automation tasks in a largely linear and user-invoked manner. Future iterations will extend this model through the introduction of an autonomous multi-agent orchestration framework based on a Planner–Actor architecture. Within this framework, a

centralized Master Agent will be responsible for interpreting high-level user objectives and decomposing them into a sequence of executable sub-tasks.

Each sub-task will then be delegated to specialized agents aligned with existing system modules. For instance, a high-level instruction such as "Prepare the monthly sales report" could automatically initiate a workflow in which the Action Recorder retrieves data from a secure web portal, the Smart PDF Writer transforms the extracted data into a formatted report, and the Smart Directory Maker archives the final output in an appropriate organizational structure. This approach enables goal-oriented automation that more closely resembles human task planning and significantly reduces the need for manual orchestration.[15]

6.2 Visual Semantic Automation (VLM)

To further improve automation robustness and adaptability, future versions of the system aim to integrate local Vision–Language Models (VLMs) such as Fuyu-8B or LLaVA. This enhancement will elevate the existing image recognition capabilities from low-level template matching to semantic visual understanding.

Rather than relying on static visual patterns or recorded coordinates, the system will be capable of interpreting instructions grounded in visual semantics, such as "click the blue 'Submit' button" or "select the confirmation dialog on the right." By reasoning over both visual and linguistic cues, the agent can dynamically identify interface elements across varying layouts, themes, and screen resolutions. This semantic approach substantially increases macro resilience and reduces failure rates caused by interface changes or display inconsistencies.[16]

6.3 Community Macro Marketplace

To promote extensibility and collaborative innovation, future development plans include the introduction of a community-driven macro marketplace. This platform will allow users to share and reuse curated automation assets, referred to as Action Recipes, as well as reusable Tone Templates for generative content modules such as the PowerPoint Maker.

The marketplace will be designed with security as a first-class concern. All shared assets will undergo peer review, strict sandboxing, and code-signing verification prior to distribution, ensuring that community-contributed macros can be safely executed on other users' systems. By enabling secure knowledge sharing, this ecosystem aims to accelerate adoption, reduce duplicated effort, and foster a collaborative environment around desktop automation best practices.

7. CONCLUSION

The Smart Macro Tool addresses a critical gap in the current automation landscape by unifying the reliability of deterministic, system-level automation with the flexibility and contextual understanding of modern artificial intelligence. Existing solutions tend to polarize between rigid, programmer-centric scripting frameworks and cloud-based intelligent agents that, while powerful, introduce significant privacy, security, and compliance concerns. The proposed system offers a balanced alternative that delivers intelligent automation without compromising data locality or user control.

By integrating precise, low-level libraries such as python-pptx and pynput with the semantic reasoning capabilities of locally deployed Large Language Models, the tool effectively tackles the "last mile" problem of digital work—those repetitive, interaction-heavy tasks that remain resistant to traditional automation. Its system-level design enables cross-application workflows, while its modular architecture supports extensibility and long-term evolution.

Looking forward, future enhancements will focus on advancing the system's visual understanding capabilities, particularly through the incorporation of Visual Question Answering (VQA) techniques. This direction will enable the agent to reason about on-screen content in a more human-like manner, further increasing the robustness of user interface automation in dynamic and visually complex environments. Collectively, these advancements position the Smart Macro Tool as a practical, secure, and intelligent foundation for next-generation personal desktop assistants.[1],[16]

REFERENCES

[1] W. M. P. van der Aalst, M. Bichler, and A. Heinzl, "Robotic Process Automation," Business & Information Systems Engineering, vol. 60, no. 4, pp. 269–272, 2018.

[2] S. Gulwani, W. R. Harris, and R. Singh, "Spreadsheet Data Manipulation using Examples," Communications of the ACM, vol. 55, no. 8, pp. 97–105, 2012.

[3] T. Vu, A. Willis, and D. Tran, "Private LLM: Architecture for Secure Document Automation," Journal of Privacy Preserving AI, vol. 4, no. 2, pp. 45-58, 2024.

[4] D. Glassman et al., "Interactive Program Synthesis by Example," in Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20), 2020.

[5] Microsoft Research, "PROSE: Program Synthesis using Examples SDK," Microsoft Research Technical Report, 2017.

[6] G. Evans et al., "Tools to Support Blind Authors in the Creation and Checking of Word Processed Documents," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 11, no. 3, 2003.

[7] Software Documentation & Standards

[8] Scanny, S., "python-pptx: Creating and updating PowerPoint files in Python," Release v0.6.21, 2023. [Online]. Available: https://python-pptx.readthedocs.io/

[9] Tauri Working Group, "Tauri: Build smaller, faster, and more secure desktop applications," 2024. [Online]. Available: https://tauri.app/

[10] Palmér, M., "pynput: Monitor and control input devices," PyPI, 2023. [Online]. Available: https://pypi.org/project/pynput/

[11] Kozea Community, "WeasyPrint: The Awesome Document Factory," 2024. [Online]. Available: https://weasyprint.org/

[12] Gerolin, P. et al., "Llama.cpp: Port of Facebook's LLaMA model in C/C++," GitHub Repository, 2024.

[13] Industry Reports & Design Patterns

[14] M. Fowler, "GUI Architectures," MartinFowler.com, 2006.

[15] UiPath, "Robotic Process Automation: A Scientific and Industrial Systematic Mapping Study," IEEE Access, vol. 8, pp. 39113-39132, 2020.

[16] Horvitz, E., "Principles of Mixed-Initiative User Interfaces," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1999, pp. 159–166.