# Educating Low Digital Literacy Users Through Agentic AI: A Multimodal On-Screen Coaching Assistant

**Yvonne Ngu Rui Ging [1], Kurk Wei Yi [2], Zahriah Othman[3], Raja Rina Raja Ikram[4], Nooraini Yusoff[5]., [5]Fakulti Sains Data Dan Komputeran**

**[1,2,3,4]Fakulti Teknologi Maklumat dan Komunikasi Universiti Teknikal Malaysia Melaka (UTeM), Durian Tunggal, Melaka, 76100, Malaysia**

**[5]Universiti Malaysia Kelantan (UMK), Pengkalan Chepa, Kota Bharu, 16100, Kelantan, Malaysia**

**DOI: https://dx.doi.org/10.47772/IJRISS.2025.908000636**

**Received: 20 August 2025; Accepted: 27 August 2025; Published: 25 September 2025**

## ABSTRACT

The increasing complexity of mobile applications poses significant challenges for users with limited technical literacy, often leading to frustration and abandonment of tasks. Existing AI-based assistants, such as Google Gemini and Microsoft Copilot, provide contextual help but do not provide detailed, step-by-step instructions specifically designed for mobile devices. This paper introduces Solus, an intelligent on-screen guidance assistant designed to provide real-time, multimodal support directly on Android devices. Unlike conventional assistants that rely on lengthy responses or task automation, Solus employs an STT-LLM-TTS pipeline integrated with real-time screen analysis to deliver interactive voice and visual step-by-step instructions. Developed using an Agile methodology, the system integrates technologies such as Google Cloud AI services, LiveKit Agent Framework, and Porcupine wake word detection. Results from prototype testing demonstrate that Solus reduces cognitive load, improves task efficiency, and empowers users to navigate digital applications independently. The findings highlight Solus's potential to foster digital inclusion and inform the design of future human-centric AI assistants.

**Keywords:** Human-Computer Interaction, On-Screen Guidance, AI Assistant, Cognitive Load, Multimodal Interaction, Mobile Usability

## INTRODUCTION

Smartphones have become essential for accessing services such as banking, education, and communication. However, the growing complexity of mobile applications often overwhelms users, particularly those with limited digital literacy, leading to frustration and task abandonment. Traditional support tools, such as search engines or chatbot-based assistants, force users to switch contexts and manually frame issues, making them inefficient for non-technical users. Moreover, while systems like Google Gemini have advanced capabilities in multimodal, context-aware assistance, they often provide bulk responses rather than interactive, step-by-step guidance in real time [1].

In the field of Human–Computer Interaction (HCI), Cognitive Load Theory advocates breaking tasks into smaller steps to reduce extraneous cognitive effort. Empirical studies support this approach in multimodal assistance contexts. For instance, dynamic visual highlighting remains effective at capturing user attention even under high cognitive load in dual-task environments, thus enhancing task performance [2]. Similarly, neuroadaptive systems that adjust guidance modalities (visual, auditory, textual) in real time based on cognitive workload have shown significant benefits in demanding settings such as aviation [3].

Yet, most intelligent assistants today are either reactive, requiring explicit user prompts, or adopt automation-first paradigms that circumvent user engagement. While convenient, these approaches limit user autonomy and hinder learning. There remains a clear need for interactive, multimodal, step-by-step guidance systems tailored for mobile devices, capable of anticipating user needs while minimizing cognitive burden.

To address this gap, this paper introduces Solus: The On-Screen Guidance Assistant, a novel system that integrates speech recognition, large language models, and real-time visual feedback to deliver context-aware, multimodal, incremental assistance. Unlike conventional assistants, Solus empowers users to complete tasks independently by providing synchronized auditory and visual cues, thereby supporting comprehension and fostering digital literacy.

## Background

Digital assistants have developed rapidly over the past decade. Early systems were mostly text-based and required users to type or search for solutions, which was not always user-friendly for people with limited digital skills. With the rise of voice technologies, assistants like Apple's Siri and Amazon's Alexa made it easier to interact with devices through natural speech. However, these tools were still limited by fixed commands and a lack of deeper context [4].

Recent advances in large language models (LLMs) have made assistants more flexible. For example, Microsoft Copilot adds AI into everyday applications to help users finish tasks faster [5]. Another example is Rabbit R1, which uses Large Action Models (LAMs) to control apps directly and carry out actions for the user [6]. While powerful, these tools can also make users too dependent on automation, since the system does everything instead of teaching the user how to do it.

A newer approach is multimodal assistance, which combines visual, voice, and contextual guidance. Studies show that using multiple modes together can reduce mental effort and improve understanding [7]. This makes multimodal systems useful in areas like education and healthcare, where clear step-by-step support is important.

## Related Work

The table below shows the related works for existing AI-driven assistance systems compared with the proposed application:

Table1 Comparison Between Existing And Proposed System

| **System** | **Strengths** | **Limitations** |
|---|---|---|
| **Google Gemini** | Provides multimodal, context-aware responses combining text, images, and voice | Responses are often bulky and lack interactive, step-by-step task guidance |
| **Microsoft Copilot Vision** | Integrates AI into productivity apps, improves workflow efficiency | Mainly designed for desktop productivity, less suitable for mobile step-by-step support |
| **Rabbit R1 (LAM)** | Automates tasks by directly controlling apps on behalf of the user | Focuses on automation rather than teaching users, leading to user dependency |
| **Proposed: Solus** | Offers interactive, multimodal, real-time on-screen guidance with voice and visual cues | Specifically designed for mobile apps, reduces cognitive load, supports autonomy and learning |

Google Gemini allows users to interact via text, voice, or image, offering powerful multimodal capabilities. However, reports indicate that its outputs are often bulk responses, which limit its usefulness for incremental task guidance on mobile devices [8]. Microsoft Copilot Vision, embedded into Windows 11 and its mobile app, delivers contextual assistance by "seeing" what's on screen like summarizing documents or identifying images. Yet, this real-time feature has triggered privacy concerns, despite Microsoft's assurances that it activates only with user consent [9].

Rabbit R1, utilizing Large Action Models (LAMs), automates tasks like ordering food or calling rides, reducing manual effort. Although innovative, reviews highlight performance issues like sluggish responses, limited reliability, and short battery life factors that undermine user autonomy [10].

In contrast, the proposed Solus system provides incremental, multimodal, and interactive support for mobile users. By integrating speech recognition, visual cues, and LLM-driven guidance, Solus reduces cognitive load while encouraging user engagement and learning.

## METHODOLOGY

The methodology adopted for this study was structured into four key phases: planning, system design, development, and evaluation. This systematic approach ensured that the proposed Solus mobile application was not only functional but also addressed the limitations of existing intelligent assistant systems.

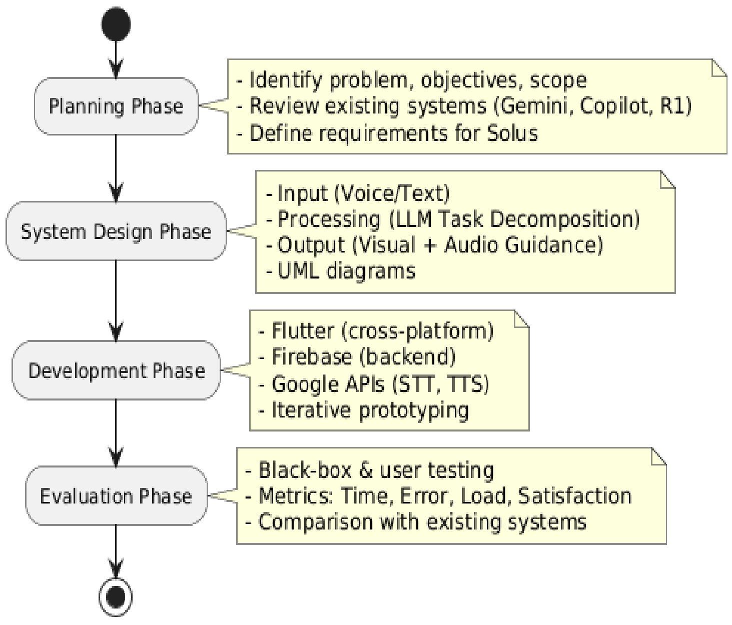

Planning Phase
- Identify problem, objectives, scope
- Review existing systems (Gemini, Copilot, R1)
- Define requirements for Solus
System Design Phase
- Input (Voice/Text)
- Processing (LLM Task Decomposition)
- Output (Visual + Audio Guidance)
- UML diagrams
Development Phase
- Flutter (cross-platform)
- Firebase (backend)
- Google APIs (STT, TTS)
- Iterative prototyping
Evaluation Phase
- Black-box & user testing
- Metrics: Time, Error, Load, Satisfaction
- Comparison with existing systems

Fig1 Methodology Phases

### A. Planning

The planning phase established the foundation for the Solus system by defining the research problem, objectives, and scope. This stage also involved conducting a review of existing multimodal assistance systems such as Google Gemini, Microsoft Copilot Vision, and Rabbit R1 to identify their limitations in providing step-by-step guidance for mobile users. From this review, the functional and non-functional requirements of Solus were specified, focusing on usability, multimodal support, and reduction of cognitive load. These requirements served as a blueprint for the subsequent design and development phases.

### B. System Design

The system design phase established the architecture of Solus and its interactions with external entities. The context diagram provides a high-level overview of the entire Solus system, representing it as a single process. It illustrates the system's interactions with external entities.

As shown in Figure 2, the user interacts with Solus by providing a wake word, voice input, and screen interactions. in return, they receive spoken response, visual cues, and conversation context. The porcupine wake word engine processes audio for wake word detection from Solus and returns a wake word detected signal. The LiveKit Cloud Server acts as a media relay, handling real-time audio, video, rpc data and real-time media & data streams between the frontend and the backend agent. Finally, Google Cloud Services provides the necessary AI capabilities, receiving audio for STT, text & images for LLM, and text for TTS, and returning transcribed text, LLM response, and synthesized speech.

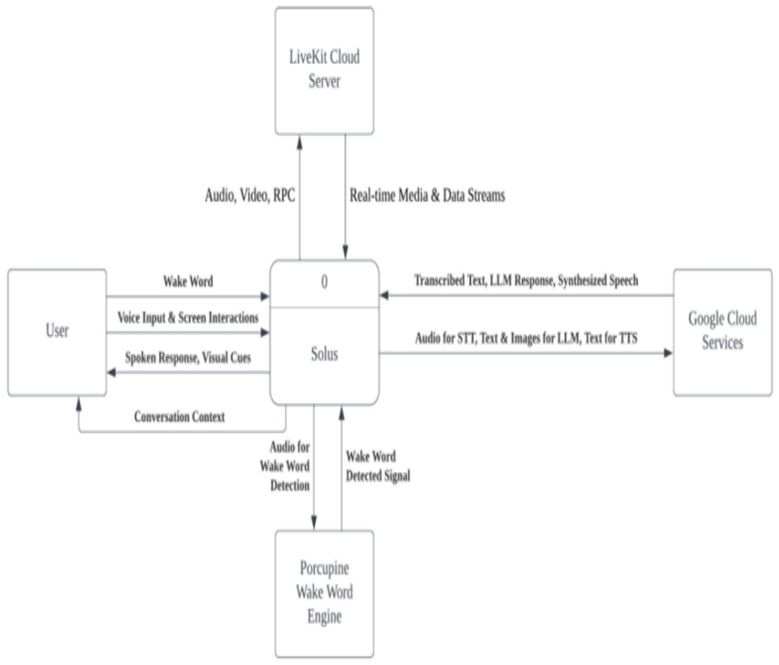

LiveKit Cloud Server
Audio, Video, RPC
Real-time Media & Data Streams
User
Wake Word
Voice Input & Screen Interactions
Spoken Response, Visual Cues
Conversation Context
0
Solus
Transcribed Text, LLM Response, Synthesized Speech
Audio for STT, Text & Images for LLM, Text for TTS
Google Cloud Services
Audio for Wake Word Detection
Wake Word Detected Signal
Porcupine Wake Word Engine

Fig2 DFD Level 0 Solus

## C. Development

The Solus mobile application was developed using Flutter for cross-platform support, while Firebase was utilized for backend services, including authentication and data storage. Speech-to-text and text-to-speech services were integrated using Google APIs, enabling seamless communication between users and the system. A modular coding approach and iterative prototyping were applied, allowing for incremental refinement based on testing feedback.

## D. Evaluation

The evaluation phase was conducted to validate the functionality and usability of the Solus application. Two approaches were employed: black box testing and user testing. Black-box testing was used to verify that each module of the system performed as intended without requiring access to the internal code. Test cases were designed based on system requirements, ensuring that core functions such as speech recognition, task decomposition, and on-screen guidance operated correctly.

## Solus

This section presents the design architecture of the Solus application and illustrates its functionality through actual screenshots of the implemented prototype. The design emphasizes seamless integration with the Android operating system, ensuring that users can interact with the assistant without switching applications or disrupting their workflow.

Figure 3 decomposes the Solus system into its three primary sub-processes, providing a more detailed view of the system's internal workings. It illustrates the three core processes: Manage User Session, Process & Contextualize Inputs, and Generate Guidance. Manage User Session is the entry point for all user interactions. It listens to the wake word via the Porcupine engine, manages the session's lifecycle, and controls the user interface (the bubble). It streams user speech and screen video to the LiveKit Cloud Server and delivers agent speech and visual cues to the user. Process & Contextualize Inputs is the central processing hub of the backend agent. It receives user speech and screen video from the LiveKit server. It orchestrates the STT-LLM- TTS pipeline by sending data to and receiving data from Google Cloud Services. It uses the current screen image and interactive UI element data to build context for the LLM. Generate Guidance is responsible for generating actionable, step-by-step instructions. It analyses the screen to identify interactive elements and then formulates the instructions that are sent back to Process 2.0 and ultimately displayed on the user's screen via Process 1.0.

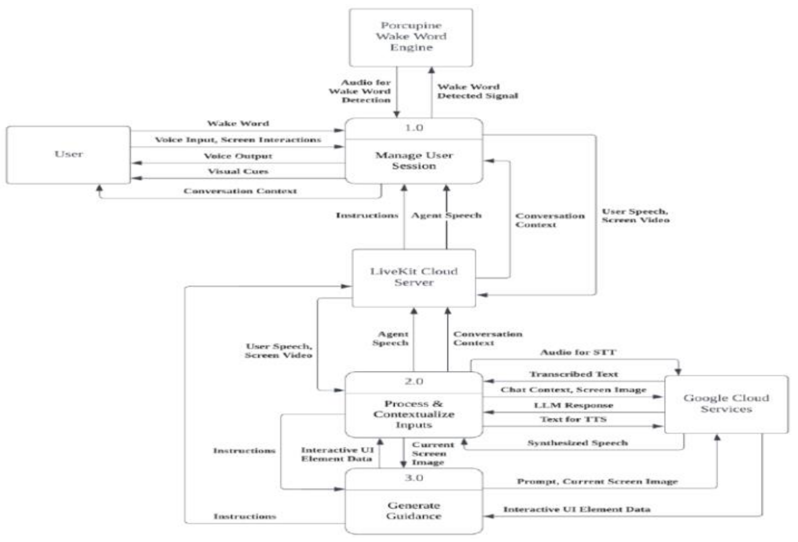
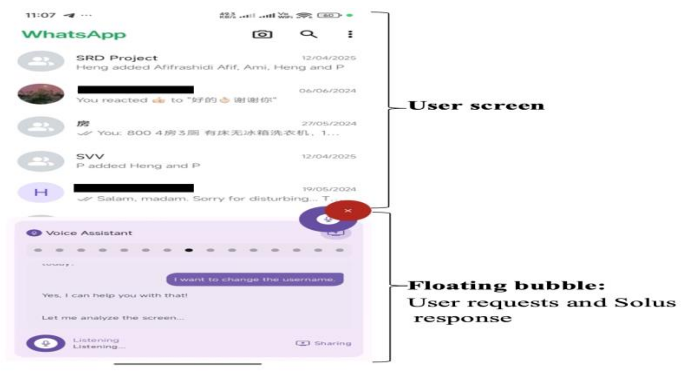


Fig3 DFD Level 1 Solus

Figure 4 shows the UI when the application is launched, Solus remains idle until the user activates it by the wake word *"Solus"*. This hands-free activation ensures convenience and avoids the need to switch between applications.



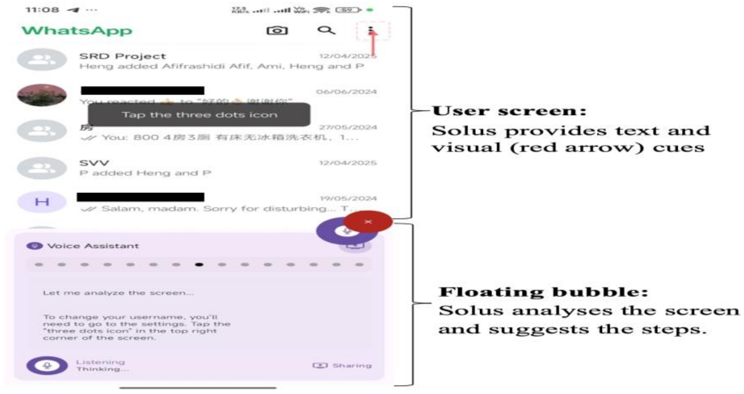
I want to change the username.

Yes, I can help you with that!

Let me analyze the screen...

Fig 4 App Launch and Floating Bubble Overlay

Once activated, a floating bubble appears on the screen as a lightweight overlay. The bubble represents the assistant's presence and provides a minimal yet persistent control interface. This design allows the user to continue using their current application without disruption, while still being aware that Solus is ready to assist.


11:08
WhatsApp
SRD Project
12/04/2025
Heng added Afifrashidi Afif, Ami, Heng and P
[redacted]
06/06/2024
Tap the three dots icon
房
27/05/2024
You: 800 4房3厕 有床无冰箱洗衣机，1...
SVV
12/04/2025
P added Heng and P
H
[redacted]
19/05/2024
Salam, madam. Sorry for disturbing... T...
Voice Assistant
Listening
Thinking...
Sharing
User screen:
Solus provides text and visual (red arrow) cues
Floating bubble:
Solus analyses the screen and suggests the steps.

Let me analyze the screen...

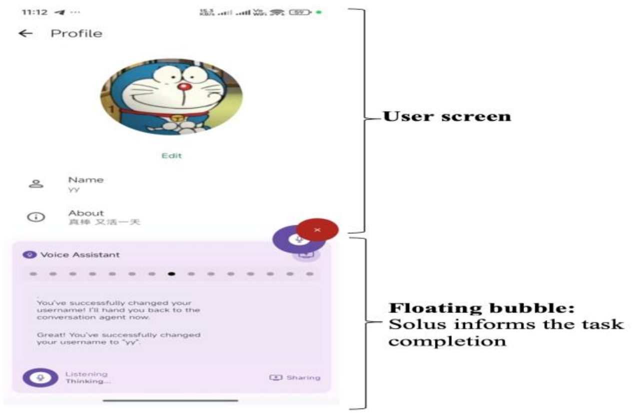
To change your username, you'll need to go to the settings. Tap the "three dots icon" in the top right corner of the screen.

Fig 5 Screen Live Analysis

Before offering guidance, Solus requests the user's permission to capture the screen using the Android Media Projection API. This step ensures user privacy and makes the assistant's functionality transparent. Once permission is granted, Solus begins analyzing the live screen content as shown in Figure 5. The captured screen frame is processed by the backend agent, which utilizes Google Gemini's multimodal capabilities to identify interface elements and understand the user's current context. This enables Solus to generate precise, context-aware instructions rather than generic responses.

Solus guides the user through tasks by combining visual and auditory instructions. In Figure 5, a specific UI element is highlighted or pointed at on the screen using a red arrow, clearly indicating where the user should act. At the same time, the system delivers a spoken instruction generated through Google Cloud Text-to-Speech. The synchronization between the visual cues and the voice instruction provides multimodal guidance, helping users to focus on one step at a time while reducing cognitive load and confusion.

When a user makes an incorrect action or deviates from the intended flow, Solus responds with corrective feedback and re-prompts them with updated instructions. This ensures that mistakes do not disrupt progress and reinforce the assistant's role as a supportive coach. Solus will guide the user until the task is completed as shown at Figure 6.



You've successfully changed your username! I'll hand you back to the conversation agent now.

Great! You've successfully changed your username to "yy".

Fig 6 Task Completion

Finally, the successful completion of the guided task is confirmed by Solus. At this stage, the assistant minimizes itself, remaining available for future activation while leaving the user with the confidence of having completed the task independently.

## RESULT

Black-box testing was conducted to verify whether the Solus application met its functional requirements. Test cases were designed for Input, Processing, and Output modules, covering both normal and invalid scenarios. The goal was to ensure that the application behaved correctly without examining the internal code structure. The results, summarized in Table 2, demonstrate that all critical functions were executed successfully.

Table 2 Black-box Testing Results

| **Test Case** | **Module** | **Description** | **Expected Output** | **Actual Output** | **Result** |
|---|---|---|---|---|---|
| TC01 | Input | Convert speech to text | Accurate text transcription | Accurate text transcription | Pass |
| TC02 | Input | Accept text command | Command received successfully | Command received successfully | Pass |
| TC03 | Processing | Interpret command with LLM | Generate step-by-step instruction | Generated step-by-step instruction | Pass |
| TC04 | Processing | Handle invalid input | Display error prompt | Display error prompt | Pass |
| TC05 | Output | Provide on-screen visual cues | Highlight relevant UI elements | Highlighted relevant UI elements | Pass |

All test cases within the modules passed successfully, confirming that the Solus application performed reliably across its core modules. Both voice and text inputs were processed accurately, the large language model (LLM) generated appropriate step-by-step instructions, and the output module provided synchronized visual and auditory feedback. Then the usability testing was conducted to evaluate how efficient and satisfying of Solus for the users where Table III shows the results.

Table 3 Usability Testing Results

| **Test Case** | **Objective** | **Metrics Captured** | **Expected Output** | **Actual Output** |
|---|---|---|---|---|
| TC01 | Measure efficiency of navigation | Time to Complete | Within designated minutes | Slightly beyond the prescribed time |
| TC02 | Identify error rates in input tasks | User Error Count.<br><br>Self-correction. | Not more than 3 errors Solus recognizes the incorrect action and provides corrective guidance | Average one error Solus corrects the user |
| TC03 | Evaluate task completion for learning activity | Task Completion Rate User Confidence: A rating from 1 to 5 | 100% successfully complete the task Self-reported confidence score is 3 or higher | 100% successfully completed the task Average user confidence score is 2 |
| TC04 | Assess satisfaction with navigation clarity | Satisfaction rating for voice clarity (1-5)<br><br>Helpfulness rating for visual overlays (1-5)<br><br>Qualitative notes from the user's open-ended feedback | Average rating for both voice clarity and visual helpfulness is 3 stars or higher.<br><br>User feedback indicates the language used by Solus is simple, polite, and easy to follow. | Average rating of 3 stars.<br><br>The voice was very clear, though the visual overlay was intermittently missing. |
| TC05 | Compare efficiency with and without assistant | Time to complete without Solus<br><br>Time to complete with Solus<br><br>Task success rate | The time taken with Solus is significantly less than the time taken without it.<br><br>The task success rate with Solus is higher compared to without Solus. | The time taken with Solus is significantly less than the time taken without it.<br><br>With Solus, 100% of the users successfully completed the task compared to 70% without Solus. |

Based on the usability testing results, the Solus demonstrated both strengths and areas for improvement. While users were able to successfully complete all learning activities and tasks, navigation proved to be slightly less efficient than expected. On the other hand, Solus reduced user errors and increased task completion rates although the visual overlay was found not appear consistently in some cases.

## CONCLUSION

The development of Solus: The On-Screen Guidance Assistant demonstrates a functional prototype that addresses information overload, inefficient context switching, and the lack of step-by-step guidance in existing assistants. By combining wake word activation, live screen analysis, and synchronized multimodal instructions, Solus reduces cognitive load and empowers users to complete tasks more confidently. Despite potential for further improvements on performance and navigation optimization, the project testing successfully validates Solus as a user-centered assistant that promotes digital inclusion and enhances accessibility for users with low digital literacy.

## ACKNOWLEDGEMENT

The authors would like to express gratitude to Centre of Advanced Communication Technology (C-ACT), Fakulti Teknologi Maklumat dan Komunikasi (FTMK), Universiti Teknikal Malaysia Melaka (UTeM) for their invaluable support and resources provided throughout this research.

## REFERENCES

1. Imran, M., & Almusharraf, N. (2024). Google Gemini as a next generation AI educational tool: A review of emerging educational technology. Smart Learning Environments, 11(22), 1–14. https://doi.org/10.1186/s40561-024-00310-z
2. Das, A., Wu, Z., Škrjanec, I., & Feit, A. M. (2024). Shifting focus with HCEye: Exploring the dynamics of visual highlighting and cognitive load on user attention and saliency prediction. Proceedings of the ACM on Human-Computer Interaction, 8(ETRA), 1–18. https://doi.org/10.1145/3655610
3. Wen, S., Middleton, M., Ping, S., et al. (2025). AdaptiveCoPilot: Design and testing of a neuroadaptive LLM cockpit guidance system in both novice and expert pilots. In Proceedings of the 2025 IEEE Conference on Virtual Reality and 3D User Interfaces (VR 2025) (pp. 656–666). IEEE. https://doi.org/10.1109/VR59515.2025.00088
4. Hoy, M. B. (2018). Alexa, Siri, Cortana, and more: An introduction to voice assistants. Medical Reference Services Quarterly, 37(1), 81–88. https://doi.org/10.1080/02763869.2018.1404391
5. Microsoft. (2024, May 21). Introducing Copilot Vision with Highlights. Microsoft Build Conference. https://blogs.microsoft.com
6. Rabbit Inc. (2024, January 9). Rabbit R1 and Large Action Models: A new paradigm for task automation. CES 2024 Launch Report. https://www.rabbit.tech
7. Chao, Y. (2025). Applications of multimodal technology in computer-assisted language learning: Impacts on cognitive load and attention distribution. Applied and Computational Engineering, 118, 107–112. https://doi.org/10.54254/2755-2721/2025.20845
8. Statt, N. (2024, May 22). An AI that sees what you see. Vox. https://www.vox.com/technology/390582/microsoft-chatgpt-copilot-vision-ai
9. Turner, A. (2025, January 16). Microsoft Copilot is putting eyes on your screen, and I don't mind it—as long as it stays private. TechRadar. https://www.techradar.com/computing/artificial-intelligence/microsoft-copilot-is-putting-eyes-on-your-screen-and-i-dont-mind-it-as-long-as-it-stays-private
10. Spoonauer, M. (2024, January 15). Rabbit R1 review: Avoid this AI gadget. Tom's Guide. https://www.tomsguide.com/reviews/rabbit-r1
11. Gupta, R., Chen, H., & Lee, J. (2025). ScreenLLM: Stateful screen schema for efficient GUI action understanding. arXiv. https://arxiv.org/abs/2503.20978
12. Nature Medicine Editorial Team. (2022). Multimodal biomedical AI. Nature Medicine, 28(9), 1717–1725. https://doi.org/10.1038/s41591-022-01981-2
13. Weichbroth, P. (2025). Usability issues with mobile applications: Insights from practitioners and future research directions. arXiv. https://arxiv.org/abs/2502.05120
14. Wang, Z.-M., Rao, M.-H., Ye, S.-H., Song, W.-T., & Lu, F. (2025). Towards spatial computing: Recent advances in multimodal natural interaction for Extended Reality headsets. Frontiers of Computer Science, 19, Article 1912708. https://doi.org/10.1007/s11704-025-41123-8
15. Bieniek, J., Rahouti, M., & Verma, D. C. (2024). Generative AI in multimodal user interfaces: Trends, challenges, and cross-platform adaptability. arXiv. https://doi.org/10.48550/arXiv.2411.10234