

The Invisible Mentor: Inferring User Actions from Screen Recordings to Recommend Better Workflows

Litao Yan
University of Pennsylvania
Philadelphia, PA, USA
ltyan@seas.upenn.edu

Andrew Head
University of Pennsylvania
Philadelphia, PA, USA
head@seas.upenn.edu

Ken Milne
Microsoft
Redmond, WA, USA
kenmilne@microsoft.com

Vu Le
Microsoft
Redmond, WA, USA
levu@microsoft.com

Sumit Gulwani
Microsoft
Redmond, WA, USA
sumitg@microsoft.com

Chris Parnin
Microsoft
Raleigh, NC, USA
chrisparnin@microsoft.com

Emerson Murphy-Hill
Microsoft
Mountain View, CA, USA
emerson.rex@microsoft.com

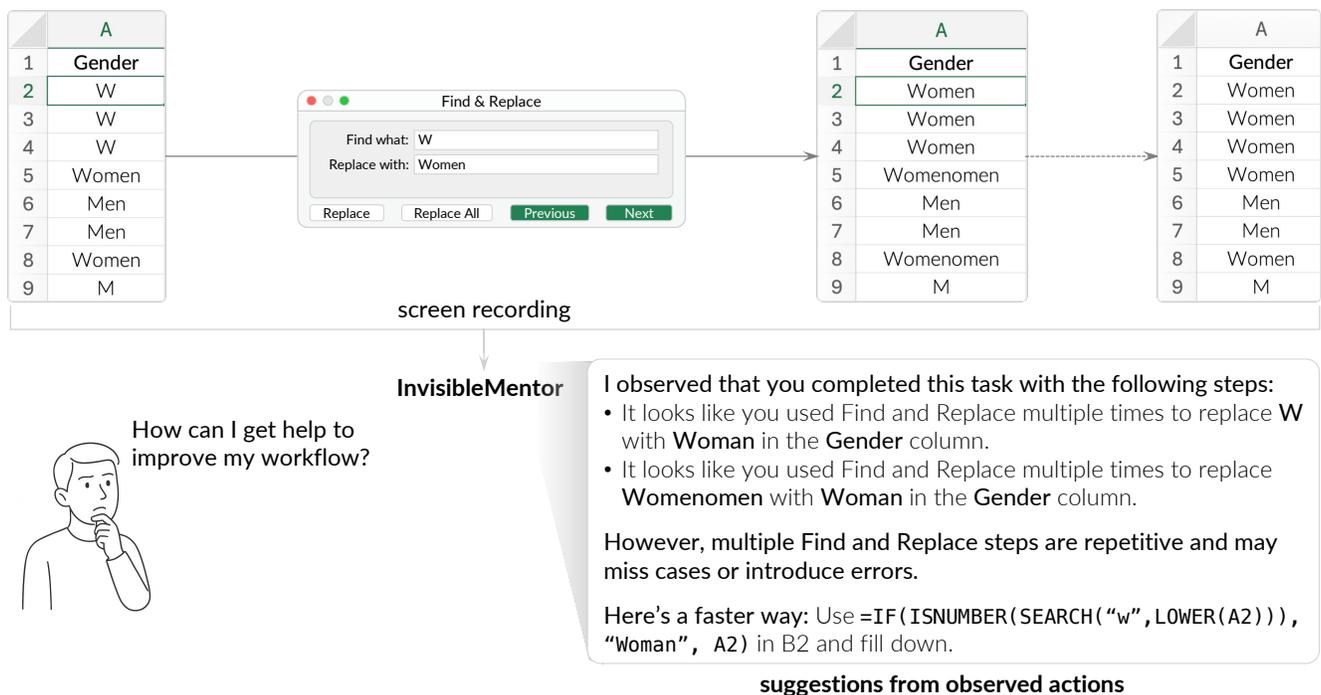


Figure 1: InvisibleMentor transforms screen recordings into expert-like mentorship. It detects inefficiencies, such as repetitive Find & Replace tasks, and synthesizes concrete alternatives. By observing visual behavior rather than requiring logs or prompts, InvisibleMentor proactively shows users better ways to work.

Abstract

Users of feature-rich tools like Excel often miss more efficient workflows, repeating tedious steps and making avoidable errors. Current approaches to helping them require either manual prompting, which is effortful for users, or automated logging, which is limiting for developers. We present InvisibleMentor, a system inspired by

over-the-shoulder learning: it observes what users do, then shows them how to do it better. To do this, InvisibleMentor analyzes screen recordings with a vision-language model to reconstruct actions and context, then uses a large language model to generate *vision-grounded task reflection*, structured suggestions grounded in observed behavior. In a user study, participants found InvisibleMentor's suggestions more clear, more relevant, and more useful than those from a prompt-based assistant, demonstrating that AI can do more than automate away work—it can help users master it.



CCS Concepts

• **Human-centered computing** → **Interactive systems and tools**.

Keywords

Intelligent Assistant, Feature Discoverability, Workflow Optimization

ACM Reference Format:

Litao Yan, Andrew Head, Ken Milne, Vu Le, Sumit Gulwani, Chris Parnin, and Emerson Murphy-Hill. 2026. The Invisible Mentor: Inferring User Actions from Screen Recordings to Recommend Better Workflows. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*, April 13–17, 2026, Barcelona, Spain. ACM, New York, NY, USA, 17 pages. <https://doi.org/10.1145/3772318.3790294>

1 Introduction

Applications like Microsoft Excel offer powerful features that support efficient work, yet most users rely on only a few basic features. For example, Figure 1 (top) shows a study participant using the *Find and Replace* feature to normalize a column of gender data—a tedious and error-prone method. Formulas or Flash Fill would simplify this task, if only the user knew when and how to use them.

Users often improve their workflows through over-the-shoulder learning [50] or peer observation [41]: an expert watches them work, spots inefficiencies, and offers advice. But such coaching is rare, limited by chance, proximity, and the availability of experts [54]. How can we make this expert guidance more frequent and accessible? We examine this question in the context of spreadsheets, where inefficient workflows are common [55].

To do so, we propose *vision-grounded task reflection*, a technique where generative AI acts as the observing peer. Our system, InvisibleMentor, implements this concept in two phases. First, a vision-language model (VLM) analyzes screen recordings to reconstruct the user’s task, including actions like inserting rows or editing formulas. Unlike prior systems, InvisibleMentor needs no system logs, internal APIs, or explicit user input. Second, a language model generates step-by-step recommendations tailored to the user’s workbook. This design produces personalized advice without requiring the user to describe their work, or even recognize their inefficiency in the first place. We built this tool to test the value of vision-grounded task reflection and to inform the design of future assistants.

This approach is attractive for three reasons. First, by recovering both action sequences and spreadsheet context, the system can generate precise, context-specific recommendations. Second, by observing entire workflows, it can uncover missed opportunities, such as overlooked features or reordering steps. Third, by shifting the burden of articulation from user to system, InvisibleMentor makes help accessible without requiring prompt crafting or intent.

We evaluated InvisibleMentor with a benchmark and a lab study. In the benchmark, our model recovered 14 common spreadsheet actions with over 90% accuracy. In a study with 20 participants, users significantly preferred InvisibleMentor over a prompt-based assistant. They found its suggestions more useful, easier to understand, and easier to apply. InvisibleMentor helped them spot inefficiencies, fix mistakes, and confidently try new workflows.

This paper contributes:

- *Vision-grounded task reflection*, an interaction mode that suggests improvements based on observed behavior, without requiring prompting or logging.
- InvisibleMentor, a system that recovers workflows from screen recordings and generates high-fidelity suggestions for efficient alternatives.
- Evidence that InvisibleMentor accurately detects inefficient workflows and helps users find better alternatives.

2 Background and Related Work

2.1 Feature Discoverability and Action-Based Suggestions

Users of feature-rich applications routinely use only a small subset of available functionality due to deep menus, terminology mismatches, and lack of guidance [18, 35, 43]. In spreadsheets, even experienced professionals tend to rely on basic arithmetic operations and manual cell references, underutilizing advanced features such as macros or pivot tables [8, 49]. Murphy-Hill et al. show that users discover new tools infrequently and largely through non-systematic channels, underscoring a persistent discoverability gap [41]. Similarly, Kohlhase et al. find that spreadsheet readers often “miss the context” intended by authors, making it hard to recognize relevant features [31].

Spreadsheets especially highlight this gap: users often rely on repetitive manual steps or overly complex formulas when simpler built-in functions would suffice [11, 49]. Even when tools exist, false affordances and ambiguous icons can mislead users as they scan for something that “looks right” [33]. Recent work further shows that discoverability barriers are not only technical but also social: Xia et al. identify how social norms and conflicted evaluations of spreadsheet expertise can limit knowledge sharing, underscoring the need for sociotechnical approaches to feature adoption [54].

To improve discoverability, prior work has proposed a range of approaches. Command search reduces hunting costs but fails when user terminology diverges from product vocabulary [19]. In-situ help provides guidance at the point of use. For example, LemonAid presents crowd-sourced Q&A linked to UI elements, allowing users to access relevant solutions without leaving the application [14], and ToolClips embeds micro-videos into tooltips for just-in-time instruction [21]. While these methods lower navigation costs, they often require users to initiate search and are not grounded in their current context or goals.

In contrast, systems that use user actions generate context-aware suggestions without explicit prompting. Early adaptive interfaces, such as Eager [15], inferred repetitive behaviors and proactively offered automation. Similarly, adaptive menus in Excel dynamically reordered commands based on usage patterns [18], showing that incorporating behavioral signals can improve efficiency while reducing cognitive load. Usage-based recommendation systems, such as CommunityCommands [39], Patina [38], and IDE fluency assistants [40], analyze historical usage patterns to uncover underused commands and identify expert practices. Complementing these, tools like DiscoverySpace [20] and Ambient Help [37] instead provide real-time, context-aware suggestions based on the user’s current task state or recent actions. In contrast, AdaptableGIMP [32]

allows users to manually browse and apply crowd-contributed workflows rather than proactively recommending actions.

More recent work advances these ideas using data-driven prediction. Wang et al. [51] analyzed millions of interaction logs and tutorial videos to recommend workflow steps matching current tasks. Nambhi et al. [42] inferred user intent from analytics tool event sequences, enabling timely suggestions when users struggle. In software engineering, Bulmer et al. [9] applied machine learning to large-scale logs of developer interactions in the IDE to predict next actions, supporting intelligent code-editing assistance. However, most of these approaches rely on structured logs or application instrumentation—limiting for developers who must build and maintain app-specific integrations [4, Chapter 3]. Our system instead infers task context and structure solely from visual evidence, enabling broader applicability without requiring internal access to software.

2.2 Recovering Task Representations from Screens

Recent work advances screen and UI understanding by recognizing elements and recovering layout structure. Vision-based models like ScreenAI [5], MobileVLM [53], and ILuvUI [27] support element targeting, instruction following, and inter-screen understanding. Other methods reconstruct user actions and workflows from screen-casts [36, 58], recovering fine-grained interaction sequences, and action-aware models map visual changes to operations [57]. Alahmadi et al. [2] extract representative UI screens from tutorial videos to segment long sessions into meaningful, task-relevant states. Similarly, analyses of “watch me code or work” videos have shown that detailed logs of user activity can be mined to understand task progress and hurdles [17].

Some systems capture detailed records of application usage through instrumentation, such as undo logs [1] or sequences of IDE commands [40]. Others infer intent in a less intrusive way by observing aggregate behavior patterns. For example, repeated scrolling or clicking activity can reveal which parts of an interface attract attention [16], and resumption cues can indicate how users re-engage with tasks after an interruption [44]. These approaches help recover implicit structure, task progress, and user struggles, offering new lenses into user workflows without manual labeling.

Domain-specific analyses show how task representations can be reconstructed in particular contexts. In spreadsheets, reasoning over cell contents, ranges, headers, and selection history is key to understanding user intent [31, 49]. Other work demonstrates how UI interactions can be extracted from videos and even translated across applications, as in ShowMeHow [45], which shows the promise of video-driven instruction recovery. These representations offer a promising foundation for AI assistants aiming to generate recommendations grounded in real user activity.

Recent work has begun applying vision-language models (VLMs) to infer user behavior directly from screen recordings. Sharingan [13] uses a VLM to label GUI actions in desktop video recordings, showing that structured action sequences can be extracted without application instrumentation. Similarly, ScreenLLM [28] transforms screen recordings into a compact stateful schema describing UI changes and uses a textual LLM to interpret or predict user actions.

In mobile domains, V2S+ [6] extracts replayable gesture scripts from Android screen videos using computer vision techniques. While promising, these systems primarily focus on action reconstruction or replay, rather than providing user-facing feedback or personalized guidance.

2.3 AI Assistants for Task-Focused Recommendations

Chat-based assistants like Excel Copilot allow users to issue commands via natural language. However, studies show that non-experts struggle to articulate goals, recall terminology, and provide adequate prompts [29, 56]. Even when functional, such assistants often lack grounding in the user’s active artifact or recent steps, leading to vague or irrelevant responses [30]. These limitations suggest the need for alternatives that observe what users do, rather than rely solely on what they say.

As an alternative, research explores recommendation methods that infer intent from user actions, anchoring suggestions in ongoing work rather than explicit prompts. Example-driven programming systems infer operations from user edits [22, 47, 48], while crowd-powered systems recommend fixes for compiler errors based on prior user behavior [23]. Mixed-initiative tools [12] propose action sequences but let users approve or modify them, balancing automation with control. Recent systems extend these directions: TableTalk uses a language agent to scaffold spreadsheet construction incrementally, yielding higher-quality outcomes with lower effort [34], and large-scale macro mining demonstrates how analyzing records of user actions can provide reusable workflows and automation opportunities [26]. Together, these approaches suggest that assistants based on users’ ongoing activity can deliver more relevant and timely guidance, avoiding the need for elaborate prompting and tying recommendations directly to what users are actually doing.

3 System Design

We designed InvisibleMentor to minimize user effort and maximize contextual relevance. Below, we articulate four goals, followed by an overview of its architecture and implementation.

3.1 Goals

G1. Translate visual representations into semantically rich representations without specialized instrumentation. Many support systems rely on internal logs or APIs to detect context and offer help (Section 2.2). While effective, these methods constrain where guidance can be offered, requiring access that is often unavailable in commercial or cross-application settings. In contrast, our goal is to extract rich, contextual representations directly from screen recordings, enabling guidance across tools with limited or no internal access.

G2. Provide structured, high-fidelity suggestions grounded in user behavior. The suggestions should not only be helpful but also intelligible and trustworthy. To achieve this, these suggestions should explicitly show what users actually did, not abstract templates, and should explain why a workflow might be inefficient, explaining missed opportunities. They should also be concrete and specific, with sufficient detail to be immediately actionable.

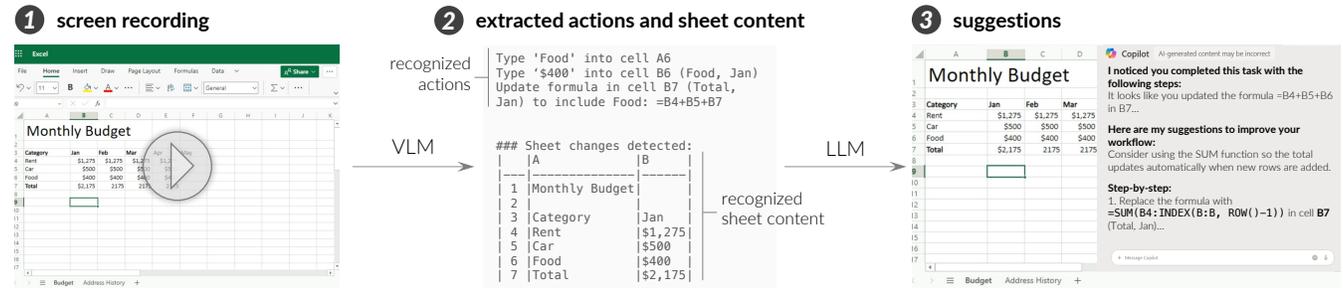


Figure 2: InvisibleMentor’s pipeline for generating suggestions from a screen recording. The system operates in two phases. (1) A vision-language model (VLM) processes screen recordings sampled every 5 seconds to extract structured task representations, including user actions and spreadsheet context (2). These representations are grouped into workflows and passed to a language model (LLM), which analyzes them to identify inefficiencies and generate actionable suggestions. Each suggestion includes a sequence of inefficient actions, a rationale, a step-by-step suggestion (3).

Together, G1 and G2 distinguish our work from prior approaches: we offer contextualized guidance without internal instrumentation by interpreting screen activity alone. These goals are complemented by additional design considerations commonly found effective:

G3. Discover unspoken help opportunities by modeling user behavior directly from observation. Most systems wait for users to ask for help (Section 2.3), but users often fail to recognize inefficiencies. As a result, help opportunities go unnoticed and underutilized. Instead, the system should proactively infer opportunities for improvement from observed behavior, like a knowledgeable peer who notices when a task could be done more easily.

G4. Deliver guidance after task completion to help users notice more efficient alternatives. We focus on post-task timing because it allows for reflection without interrupting ongoing work. Prior work shows that mid-task interventions can disrupt focus and increase cognitive load [10]. Offering guidance after the task therefore provides a lightweight moment for helping users become aware of better workflows they might adopt next time.

3.2 Implementation

We implemented InvisibleMentor as a two-phase pipeline that combines visual interpretation of user activity with textual reasoning for recommendations (Figure 2). The first phase uses a vision-language model (VLM) to capture a structured representation of the user’s task from screen recordings (Figure 2, 1), while the second phase employs a large language model (LLM) to generate grounded and actionable suggestions. We describe each phase in detail below.

3.2.1 Phase 1: Capturing Task Context with a VLM. The first phase constructs a representation of the user’s task based on visual observation (G1). As the user works in Excel, the system samples one screen frame every five seconds to capture a lightweight sequence of their on-screen actions. This cadence balances temporal coverage with efficiency: it is frequent enough to capture meaningful actions, such as formula edits, filter changes, or table creation, while avoiding redundant frames that increase cost and latency.

Task representation. Each frame encodes the spreadsheet state and interface context, including selected cells, visible formulas, column headers, named tables, and open panes. To analyze these

frames, we use GPT-4.1¹, which supports both text and image input and provided state-of-the-art image understanding during the period of development. The model produces a structured representation of user activity with two elements: (a) *Action sequence*, a list of low-level user actions (e.g., editing a formula, applying a filter, switching sheets); and (b) *Contextual state*, the spreadsheet content and layout at each moment, such as selected ranges, cell values, and interface elements (Figure 2, 2).

Because the model’s context window can accommodate only about 20 frames, roughly 100 seconds of video at our chosen interval, a long recording cannot be processed in a single pass². To address this limitation, we divide each recording into up to three temporal segments, each capturing a portion of the task. These segments are processed in parallel to reduce post-task latency. Within each segment, we further divide the frames into batches of 20 frames, each covering approximately 100 seconds of video. These batches are fed sequentially into separate VLM calls. This hierarchical strategy enables the system to recover a complete task trace from longer recordings, while respecting the model’s input limits and preserving continuity within and across segments.

Pilot testing and parameter choices. To select appropriate parameters, we conducted a pilot test using the *Europe Bike Store Sales* dataset, in which the authors performed three representative spreadsheet tasks. The entire session was recorded, producing a one-hour video for VLM testing. We compared different sampling intervals and segmentation strategies. At a 1-second interval, the VLM repeatedly detected near-duplicate actions because a single user action typically spanned several seconds; this redundancy inflated processing time and yielded little additional information. In contrast, a 10-second interval often skipped important, transient behaviors, such as brief pop-up messages or rapid double-clicks to apply formulas. Balancing coverage with efficiency, we chose a 5-second interval, which avoided both excessive duplication and missed actions.

¹All experiments used the Azure OpenAI Service deployment of GPT-4.1 (April 2025 release).

²While GPT-4.1 supports up to 1M tokens in Azure, practical limits depend on deployment type and prompt overhead. In our configuration (128k managed deployment, structured prompts, and metadata per frame), the effective budget allowed for about 20 1080p frames per call.

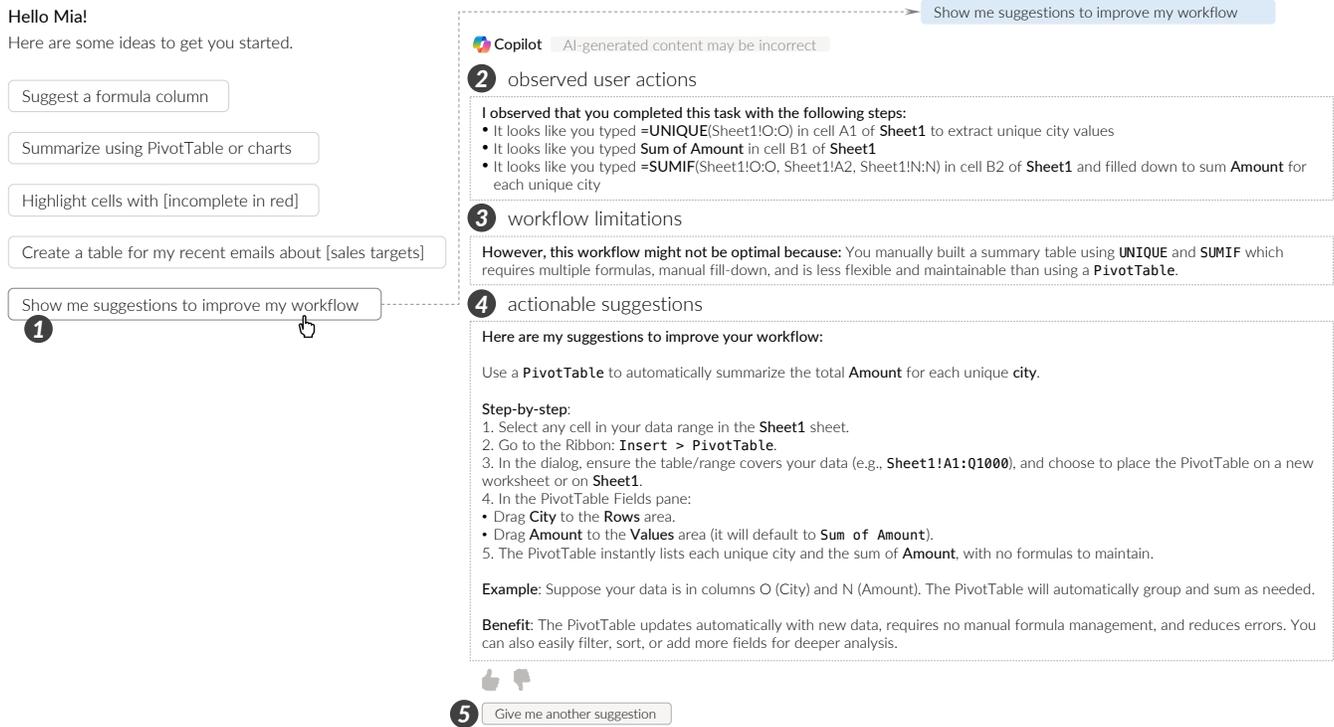


Figure 3: User interface of a spreadsheet assistant that provides structured workflow guidance. The assistant appears in a task pane alongside the spreadsheet, following the familiar layout of Excel Copilot to minimize design variance that could influence user study outcomes. The interface consists of five key components. (1) Prompt ideas offer users high-level suggestions to initiate help-seeking. Selecting an idea appends a message to the conversation and triggers a structured assistant response. To make accessing suggestions more direct, we replaced the original fifth idea with a dedicated entry point for requesting recommendations. The remaining components are rendered dynamically based on model output: (2) observed user actions, summarizing recent spreadsheet activity; (3) workflow limitations, explaining potential inefficiencies; (4) actionable suggestions, presenting step-by-step improvements; and (5) a “give me another suggestion” button, allowing users to request alternatives.

We also compared processing times with and without segmentation. Without segmentation, processing a one-hour video at a 5-second interval required 6 minutes, and nearly 41 minutes at a 1-second interval—approaching the duration of the video itself. With parallel processing, where the video is split into three chunks that are processed simultaneously, processing time dropped to around 2 minutes at the 5-second interval³. We discuss the limitations of this sampling strategy in Section 7.1.

3.2.2 Phase 2: Generating and Delivering Recommendations with an LLM. The second phase generates structured and high-fidelity suggestions from the extracted task representation (G2 and G3). Whereas Phase 1 relied on GPT-4.1’s vision capabilities, Phase 2 uses the same model in textual mode, taking actions and contextual states as input under a zero-shot prompt (see Appendix A).

Recommendation generation. The prompt instructs the model to group related actions into workflows, mark those that are suboptimal, and return at most three prioritized recommendations. Each recommendation contains the relevant actions and a rationale for

³This estimate reflects our pilot test on a fixed one-hour recording using parallel processing. Actual latency varies with video duration, as reported in Appendix C.2, which reports processing times without parallel processing.

why they are inefficient, together with a concrete alternative (including explicit steps or formulas) and a brief “benefit” statement comparing before and after (Figure 2, 3).

Rather than streaming partial results, InvisibleMentor processes the set of observed steps from Phase 1 before generating recommendations. This design enables the LLM to detect not only local inefficiencies but higher-level repetitive patterns. For example, while a sequence of copy-paste operations may be inefficient at the action level, broader repetition often occurs at the workflow level, such as repeatedly cycling through data cleaning, processing, and analysis. Within each cycle, individual actions may appear reasonable, but when viewed together they reveal opportunities for more efficient workflows. By reasoning over the full sequence of observed user actions, the system can thus provide suggestions that address both fine-grained operations and broader workflow structures.

3.2.3 Interface. The interface (Figure 3) appears after task completion (G4). It is displayed in a task pane alongside the spreadsheet, following the familiar layout of Excel Copilot. This design minimizes interface-related confounds, helping ensure that observed

effects in the study reflect the system’s recommendations rather than novel UI elements⁴.

Prompt ideas as entry points. To reduce visual differentiation from the original Excel Copilot interface, we replaced the fifth idea card with a dedicated entry point labeled “Show me suggestions to improve my workflow” (Figure 3, ❶).

Observed user actions. The assistant summarizes the user’s recent activity in natural language (Figure 3, ❷), noting workflows identified as potentially inefficient (G2). Each description reflects a real sequence of user actions, e.g., building a summary table using UNIQUE and SUMIF, grouped into meaningful task units. These summaries serve two purposes. First, they help users recall and recognize inefficiencies that may have gone unnoticed during task completion, supporting accurate memory alignment. Second, by explicitly surfacing these patterns, the assistant builds credibility for subsequent suggestions, allowing users to judge whether its interpretation aligns with their memory and experience.

Workflow limitations. The system explains potential inefficiencies in the observed workflow (Figure 3, ❸), such as using multi-step formulas where built-in features would suffice. These explanations help users understand not just what could change, but why.

Actionable suggestions. Recommendations are presented as concrete, step-by-step instructions (Figure 3, ❹), tailored to the user’s workflows (G2). Each suggestion explicitly names the relevant features (e.g., Get Data, Merge Queries) and provides exact Ribbon paths or shortcuts for accessing them. When applicable, the system also includes output examples to help users verify their results. Suggestions are designed to be executable within the spreadsheet environment, even for advanced features like Power Query or PivotTables. Each ends with a concise benefit statement that compares the original workflow to the recommended one, showing time savings, reduced errors, or improved maintainability (e.g., “Original: 4+ steps, Suggested: 2 steps”). This framing encourages adoption by making the value of the recommendation clear and tangible.

Alternatives on demand. To reduce information overload, the assistant initially presents only the most critical recommendation. Users can request additional suggestions using the “Give me another suggestion” button (Figure 3, ❺), which sequentially reveals other suboptimal workflows identified during analysis. This interaction model enables a lightweight, controlled exploration of the full recommendation set, allowing users to focus on one improvement at a time without becoming overwhelmed.

4 Technical Evaluation

A core requirement of our system is the ability to reliably recover users’ spreadsheet actions from screen recordings. Because suggestion quality depends directly on the fidelity of this reconstructed action sequence, we first evaluate the reliability of InvisibleMentor’s VLM-based action-recognition component. Before assessing the quality of generated suggestions in Section 5, we address the following research question in this technical evaluation:

RQ1. How accurately can a VLM identify users’ spreadsheet actions from screen recordings?

4.1 Methodology

Dataset. To evaluate RQ1, we used an internally collected benchmark dataset of spreadsheet interaction videos. The dataset contains 25 screen-recorded sessions where participants, recruited via Playbook UX, completed 14 short tasks involving basic yet practical Excel operations, such as inserting rows, editing formulas, switching sheets, and renaming tabs. Each session used an Excel workbook with two pre-filled sheets, each containing 6 to 12 rows of synthetic data⁵. Each task is an isolated atomic interaction sequence typically consisting of one to three user actions, enabling a fine-grained assessment of recognition performance in spreadsheet environments. We were not involved in its original collection but used it for evaluation.

Procedure. Before evaluation, we preprocessed all videos by cropping out interface regions that displayed task instructions or unrelated UI elements, and sampling one frame every five seconds to generate inputs for the VLM. This prevented instruction leakage and ensured the model inferred actions from observed behavior rather than visible text. The VLM then produced an ordered sequence of predicted user actions for each session, together with contextual cues derived from the spreadsheet interface.

To establish ground truth, we manually reviewed every video to determine whether each task was completed, attempted but not finished, or skipped due to time or user choice. These binary annotations (performed vs. not performed) served as the reference labels for evaluating action recognition.

Evaluation. We evaluated the VLM’s predictions against the annotated ground truth using accuracy, precision, recall, and F1 scores. Since each task in the dataset corresponds to an atomic spreadsheet operation (e.g., inserting a row, switching sheets, editing a formula), these metrics reflect the model’s ability to recognize individual user actions rather than broader task success. To assess reliability of the ground truth, we also calculated the percent agreement and Gwet’s AC1 to measure agreement between human annotations and an independent LLM (GPT-4.1) in labeling the VLM’s outputs. Finally, to better understand the model’s limitations, we analyzed error distributions across task types, identifying which spreadsheet operations the VLM most frequently failed to detect.

4.2 Results

Recognition accuracy. We evaluated the VLM’s predictions by comparing its inferred action sequence against the annotated ground truth for all 350 task instances. The model achieved an average F1 score of 0.91 (human rater) and 0.90 (LLM rater). Most sessions (21 out of 25) achieved F1 scores above 0.85, indicating consistent performance across users. We observed no false positives (FP = 0) in the dataset, showing that the VLM did not hallucinate actions. Therefore, all recognition errors came from actions the model failed to detect.

⁴See our user scenario in Appendix B.

⁵See Appendix C.1 for a detailed list of tasks and spreadsheet content used in the technical evaluation.

Inter-rater agreement. To assess consistency between the two raters, we computed percent agreement and Gwet’s AC1. The raters agreed on 96.4% of action-level decisions ($\pm 5.4\%$), with a Gwet’s AC1 of 0.927 (± 0.107), indicating strong agreement beyond chance.

Error analysis. Because all recognition errors involved actions the model failed to detect, we examined where these failures occurred. Most missed actions involved operations that took place between sampled frames or subtle interface changes that were difficult to capture. To better understand these patterns, we grouped task types into (1) functional workflow actions that advance task progress and (2) formatting operations that primarily alter visual appearance. Functional workflow actions, such as entering values, editing formulas, switching sheets, opening files, creating new sheets, copying and pasting, applying fill color, and sharing the workbook, were detected consistently and showed no missed instances in our dataset. A few task types showed moderate error rates, such as adding a row (12.5%), creating a new workbook (17.4%), and renaming sheets (5.0%). Missed actions were concentrated in formatting operations, including bold text (58.3% missed), font-color changes (35.7%), and column resizing (18.8%). These operations often introduce only subtle visual changes, such as slight differences in text weight, color, or cell boundaries, which makes them less salient in the sampled frames and explains their higher omission rates.

Overall, the VLM recognized functional workflow actions with high consistency, while missed actions were concentrated in subtle formatting changes. These omissions may prevent InvisibleMentor from making certain appearance-related suggestions, but they do not compromise the higher-level workflow reasoning that our recommendations rely on. We therefore consider the action-recognition performance sufficient for supporting the types of recommendations evaluated in Section 5, while noting opportunities for improving fine-grained formatting detection.

5 User Study

To evaluate the effectiveness and user experience of InvisibleMentor’s recommendations, we conducted a within-subjects user study. Participants completed spreadsheet tasks using both InvisibleMentor and a strong baseline: Excel Copilot, a prompt-based assistant purpose-built for spreadsheet support. Unlike general chat interfaces, Excel Copilot has direct access to spreadsheet content and is designed to generate context-specific recommendations based on user queries, making it a high-performing point of comparison.

Our study sought to evaluate whether InvisibleMentor’s approach, observing user behavior through screen recordings and offering high-fidelity suggestions based on interpreted action traces, can help users discover and reflect on opportunities to improve their workflows. We organized our investigation around the following research questions:

RQ2. Can the system accurately identify inefficient or suboptimal workflows by observing user behavior alone?

A core goal of our system is to detect inefficient workflows without relying on internal instrumentation (e.g., logs or APIs) or user prompts. We designed InvisibleMentor to analyze screen recordings to provide targeted suggestions derived from semantic traces of user actions after completing the task (G1, G3). This question asks

whether the system succeeds in identifying real inefficiencies in users’ task execution based solely on visual observation.

RQ3. How do users interpret and respond to suggestions grounded in their actual behavior?

We aim to generate suggestions that are not only accurate but also comprehensible and actionable. Suggestions should reflect what the user actually did, explain why the approach may be inefficient, and provide concrete alternatives (G2, G4). This question probes whether users find these behavior-grounded suggestions clear, relevant, and actionable.

RQ4. How does visual-grounded task reflection compare to prompt-based assistance in providing helpful guidance?

In contrast to traditional AI assistants that wait for user input, InvisibleMentor proactively identifies help opportunities by interpreting users’ actions. This question evaluates whether this behavior-grounded approach reduces user effort compared to prompt-based alternatives, and whether it helps users notice more efficient workflows they might not have discovered on their own.

5.1 Methodology

5.1.1 Participants. We recruited 20 participants through the Playbook UX platform. Participants spanned a wide range of occupations, including analysts (7), administrative staff (6), managers (4), an accountant (1), an engineer (1), and a student (1). Their ages ranged from 23 to 56 years ($\mu = 34.4$, $\sigma = 9.3$). The sample included 12 participants who identified as male and 7 as female⁶.

Participants reported regular use of spreadsheet tools in their work: 75% (15/20) used Excel or other spreadsheet software daily, while the remaining 25% (5/20) used it weekly. In terms of experience, 50% had more than 10 years of experience, 25% had 6-10 years, and 25% had 3-5 years.

We also asked about AI usage in spreadsheet tasks. While 45% (9/20) of participants reported never using AI for spreadsheets, 35% (7/20) used AI tools weekly, and 10% (2/20) used them daily. The most commonly mentioned tools were ChatGPT and Copilot. Other tools included Gemini, Claude, and Excel’s built-in Ideas feature.

5.1.2 Procedure. Each participant completed a one-hour study session. After providing informed consent, participants first filled out a background questionnaire about their prior experience with Excel. To minimize potential bias, we referred to tools using neutral names (“Tool A” for InvisibleMentor and “Tool B” for Excel Copilot) and counterbalanced the order where participants used them. The session proceeded through three phases: a data challenge task, a tutorial for both tools, and an evaluation session in which participants reviewed and assessed recommendations from both tools. We chose not to show suggestions during the tasks to avoid disrupting participants’ focus or altering their natural workflows. Instead, suggestions were shown only after task completion to help participants recognize where their workflows could be improved, aligning with our goal of providing just-after-it-mattered guidance (G4).

Data Challenge Task. Participants completed one of two realistic Excel-based data challenges designed to simulate multi-step workflows in large, multi-sheet workbooks. The design of these

⁶One participant did not report gender.

challenges was inspired by Kaggle-style analytics problems, where participants work with realistic datasets to answer open-ended analysis questions. Each challenge contained four questions involving tasks such as summarizing trends, analyzing category-based performance, and identifying opportunities for improvement (see Appendix D for task details). To maximize coverage of different spreadsheet operations, each participant completed only one of the two challenges, and task assignment was counterbalanced across participants. Pilot testing with five participants showed that the two challenges were matched in both completion time and perceived complexity. Each question in the challenge could be completed using either simple but repetitive operations (e.g., filtering, sorting, and basic formulas) or more advanced features such as SUMIF formulas and PivotTables, enabling us to evaluate system effectiveness across diverse workflows and experience levels. During the task, the experimenter recorded the entire window shared by the participants for later analysis.

Tutorial. Before the suggestion evaluation, participants were given five minutes to familiarize themselves with each tool through a tutorial. In the baseline condition, participants typed their descriptions of recent steps and context into a text input box. In contrast, with InvisibleMentor, participants clicked the “Show me suggestions to improve my workflow” button (Figure 3, ❶) to receive the first suggestion, and could then use the “Give me another suggestion” option (Figure 3, ❷) to request additional suggestions (three suggestions in total). The features introduced during this tutorial were identical to those available during the evaluation session.

Suggestion Evaluation Session. After completing the data challenge tasks, participants entered a recommendation evaluation session where they used both tools to retrieve, review, and optionally test three recommendations per tool. The order of tools was counterbalanced. In the baseline condition, participants were instructed to prompt the tool to generate suggestions. In the InvisibleMentor condition, recommendations were generated automatically from participants’ screen recordings and displayed by the experimenter in the Excel interface. Screen recording was initiated by the experimenter at the start of the task to support VLM processing, but participants were informed that in a fully integrated version of InvisibleMentor, this would occur automatically. For each suggestion, participants could decide whether to test it on the same spreadsheet used in the earlier task. We logged whether they attempted the suggestion, and whether the suggested steps, when followed, produced the correct outcome.

Questionnaires. Participants completed three sets of questionnaires⁷. At the start, a brief background questionnaire collected participants’ prior experience with Excel. During the recommendation evaluation session, after reviewing each suggestion, participants rated it on eight 5-point Likert-scale items capturing multiple aspects of the suggestion (e.g., usefulness, clarity, relevance; see supplemental materials for full list). At the end of the session, a final questionnaire captured participants’ overall impressions, including perceived workload, effort required to obtain suggestions, privacy concerns, and preferences between tools.

5.1.3 Measurements and Analysis. After each suggestion, participants provided 5-point Likert ratings on eight dimensions. We analyzed these ratings using linear mixed-effects models, with fixed effects for tool, task, tool order, whether the participant attempted to apply the suggestion, whether the suggestion produced the correct result when tested, and the number of distinct tools referenced in the suggestion. A random intercept was included for participant ID. Significance was assessed using F-tests with Satterthwaite’s approximation of degrees of freedom [46].

For the three summary-level questions on overall impressions, perceived privacy confidence, effort required to obtain help, and system preference, we used two-tailed Wilcoxon signed-rank tests [52]. All p -values were adjusted using the Holm-Bonferroni method [24], with a significance threshold of $\alpha = 0.05$.

We also conducted a thematic analysis [7, Chapter 5] of participants’ open-ended responses to identify recurring themes around perceived strengths, limitations, and ideal usage scenarios.

5.2 Results

5.2.1 RQ2: InvisibleMentor identifies real inefficiencies.

InvisibleMentor detected inefficient or error-prone workflows from screen activity. Most participants ($P1, P2, P7-8, P11-14, P20$) described InvisibleMentor’s suggestions as tightly linked to their own actions, contrasting it with the baseline’s responses. They found that InvisibleMentor knew “what I was trying to do” ($P8$), “tracked what I performed” ($P13$), and “highlighted the things I did manually” ($P2$). $P1$ noted that InvisibleMentor “went based off of my attempts and edited it as I was attempting it,” while $P11$ suggested enabling mid-task delivery to make suggestions even more timely.

InvisibleMentor uncovered several commonly recurring inefficiencies across participants’ workflows, including: repeatedly using Find and Replace to standardize inconsistent categorical values like gender ($P2, P11, P19$), manually writing separate SUM formulas for each month instead of using SUMIF or PivotTables ($P1, P5, P11, P20$), and frequently copying tables between spreadsheets to apply formulas like VLOOKUP ($P12, P15, P18$).

5.2.2 RQ3: Users find behavior-grounded suggestions comprehensible and actionable.

InvisibleMentor provided clear, easy to follow, and confidently actionable suggestions. Participants found InvisibleMentor’s suggestions significantly easier to understand than those of the baseline ($F = 25.1, p < 0.001$)⁸, and reported greater confidence in knowing how to carry out the recommended actions ($F = 29.3, p < 0.001$). This perceived clarity was reflected in participants’ comments: $P11$ described the suggestions as “short, concise, and straight to the point,” while $P5$ noted they were “easier to follow.” In contrast, $P10$ found the baseline “too complex to execute,” and $P20$ remarked they “didn’t know if [baseline suggestions] were true or not.”

InvisibleMentor’s suggestions reflected the user’s recent actions and intent. Participants rated InvisibleMentor’s suggestions as significantly more relevant to their just-completed tasks ($F = 15.7$,

⁷Full questionnaires are provided in the supplemental materials.

⁸One participant did not provide a rating for the item “understand suggestion clearly” on one of the suggestions.

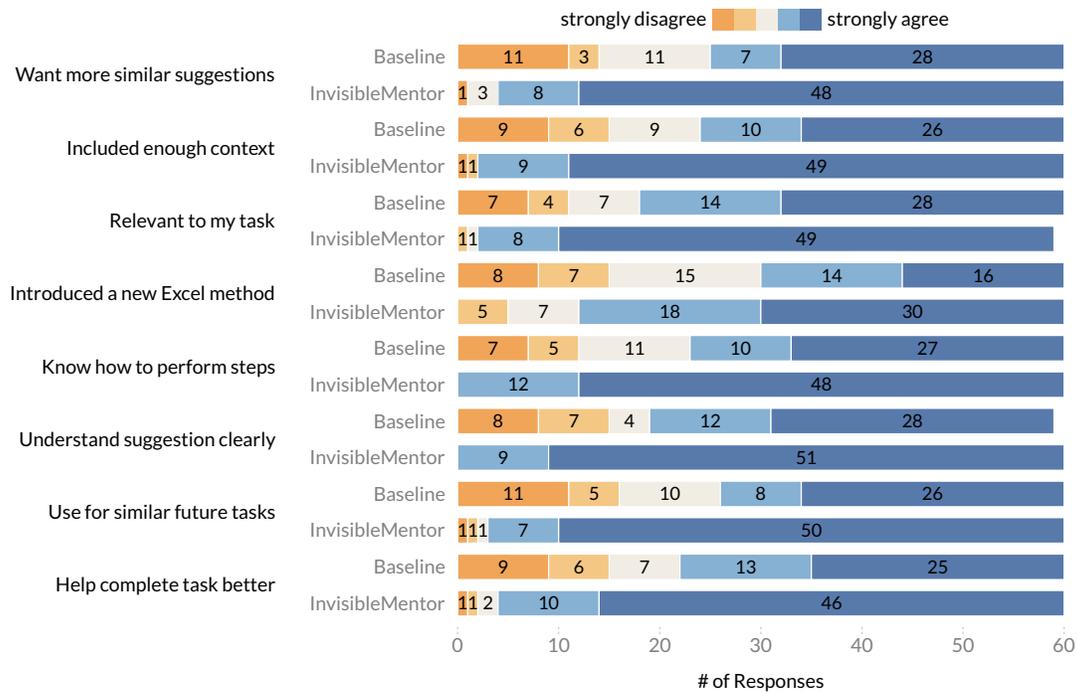


Figure 4: Participant ratings of InvisibleMentor’s suggestions. Stacked bar charts show agreement levels with eight evaluative statements on a 5-point Likert scale (from “Strongly Disagree” to “Strongly Agree”). Across all statements, participants rated InvisibleMentor’s suggestions as significantly more useful, understandable, and better aligned with their recent tasks than those from the baseline system.

$p < 0.001$)⁹, and reported receiving more contextual explanations for why a suggestion was offered ($F = 27.6, p < 0.001$). P1 said that the tool “gave me corrections based on my past attempts,” while P8 noted that it was “actively guessing what I’m trying to do to make a suggestion.”

5.2.3 RQ4: Proactive behavior-based suggestions reduce effort and help users become aware of more efficient alternatives.

InvisibleMentor reduced users’ effort compared to prompt-based alternatives. Participants rated InvisibleMentor as requiring significantly less effort to use than the baseline ($W = 0.0, p < 0.001$), and expressed an overall preference for it ($W = 8.0, p < 0.001$). Despite the system’s use of screen recordings to track user behavior, participants did not report heightened privacy concerns ($W = 27.0, p = 0.58$) compared with baseline.

InvisibleMentor’s suggestions helped users notice opportunities for more efficient workflows. Participants rated InvisibleMentor’s suggestions as significantly more useful for improving their workflows than those from the baseline ($F = 18.8, p < 0.001$). They reported greater willingness to reuse InvisibleMentor in real-world settings ($F = 23.7, p < 0.001$), and expressed interest in receiving more similar suggestions in future work ($F = 18.6, p < 0.001$) than baseline. Participants also reported that the suggestions showed them how to perform Excel actions they were previously unfamiliar with ($F = 11.6, p = 0.01$), helping them notice opportunities to work more efficiently.

⁹One participant did not provide a rating for the item “suggestion relevancy” on one of the suggestions.

Several participants described the tool as acting like a tutor (P5, P13, P15, P17, P19), offering context-sensitive feedback on their workflows and helping them notice opportunities to work more efficiently (P1-2, P8, P11, P13, P15, P17, P19). P5 remarked that the tool “taught me more and quicker,” and P19 said it helped them “learn and apply these corrections.”

In contrast, some participants described the baseline as limited in helping them identify where their workflows could be improved (P3, P5-6, P8-10, P17, P19). P10 remarked that while the baseline “felt more advanced,” its suggestions were “too complex to execute,” limiting their value. P6 observed that the baseline “would give you a suggestion dependent on what you ask.” P17 noted that the baseline was “better for getting work out there quickly,” but less useful for noticing alternative ways of working. While most participants saw InvisibleMentor as stronger for recognizing opportunities to improve their workflows, some also acknowledged the benefits of the baseline (P4, P8, P14, P18). P4 shared that the baseline “allowed me to personalize the feedback and support,” and appreciated that both systems “highlighted things I wouldn’t have initially thought about.” P18 similarly described the baseline as helpful for “a quick answer and fix.”

Participants also viewed baseline suggestions as one-off fixes (P9, P17-19). P9 noted that while baseline “would help me now,” it was InvisibleMentor that “would help me for the future.” P19 similarly reflected that baseline “had a tendency to try and do things for me,” which “may be helpful in the moment but maybe I would be less likely to apply the suggestions in the future.”

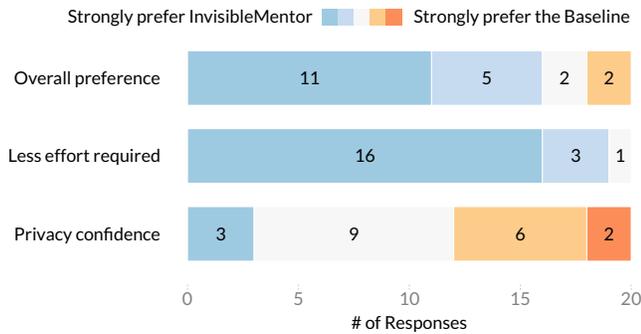


Figure 5: Participants' comparative preferences between InvisibleMentor and the baseline. Participants were asked which tool required less effort, which they would overall more prefer to use in the future, and which they trusted more with regard to privacy. Most participants significantly preferred InvisibleMentor for effort and rated it as overall more preferable to use, with no significant difference in perceived privacy confidence.

6 Discussion

6.1 Revisiting Our Study Questions

InvisibleMentor accurately recovered user actions from screen recordings using a VLM, demonstrating strong agreement with ground truth annotations (Section 4.2, RQ1). By analyzing users' behavior, it discovered inefficient or repetitive workflows such as repeated Find & Replace operations or redundant manual summaries (Section 5.2.1, RQ2). Participants described InvisibleMentor's suggestions as easier to understand and more grounded in their recent actions than baseline assistance, and quantitative ratings confirmed higher clarity, actionability, and perceived contextual relevance (Section 5.2.2, RQ3). Compared to the prompt-based baseline, InvisibleMentor required significantly less effort to use. Participants rated its suggestions as more useful for improving workflows and helping them notice unfamiliar Excel features. They also expressed stronger overall preference for InvisibleMentor and greater interest in using it again in real-world settings (Section 5.2.3, RQ4).

6.2 Evaluating recommendation usefulness and executability

To assess the usefulness and reliability of InvisibleMentor's generated recommendations, we conducted a detailed post-hoc evaluation focusing on their executability, efficiency, and error correction.

Executable validity. The authors manually executed every suggested workflow step on the original spreadsheet data used in the study. Each suggestion was reproduced exactly as written, without adding, omitting, or modifying any steps. Among the 60 evaluated suggestions, 58 successfully completed the intended workflows, achieving the same outcomes as participants' original goals. Only two suggestions produced incomplete workflows due to missing details (e.g., "use the Find & Replace feature (Ctrl+H) to update headers across the sheet if needed," which did not specify the replacement content; and "after using PowerQuery, load the result back to Excel as a new table," which omitted import parameters).

Interaction efficiency. We next analyzed the interaction efficiency of each suggestion in terms of steps saved compared with the original workflows. Step counts were measured at the granularity of atomic user interactions (each click, drag, or typing action). Across all 58 successfully reproduced suggestions, the system reduced the number of required interactions by an average of 32.4% (SD = 74.5%), with most improvements falling between 40-80%. The large variance was mainly driven by two groups of cases: (1) eight suggestions where participants stopped mid-task after realizing their workflow was inefficient, leading to conservative improvement estimates; and (2) two suggestions that intentionally introduced additional steps to promote robustness and generalizability (e.g., using named ranges or more stable formulas). After excluding these ten cases, the remaining suggestions achieved an average step reduction of 46.4% (SD = 30.7%), indicating that InvisibleMentor's recommendations achieved the same goals with substantially less interaction effort.

Error correction and robustness. Finally, InvisibleMentor identified and corrected 18 workflow errors across participants. The most frequent issues involved invalid chart or PivotTable references, where users manually typed or adjusted data sources, attempted to select non-contiguous ranges, or referenced entire columns, triggering "data source not valid" errors or blank sheets (P1-3). Several participants also produced incorrect or incomplete formulas, such as misusing the TEXT function for hour extraction (P4), providing invalid references in FORECAST or SUMIF formulas (P15, P20), or leaving computed columns partially unfilled (P12). Others engaged in manual and error-prone aggregation, repeatedly filtering or summing subsets by hand (P4-5, P11), which failed to update automatically when data changed. A few performed destructive find-and-replace operations (e.g., replacing "W" with "Women" and unintentionally producing "Womenomen," P7, P19), or applied incorrect formatting (e.g., date columns displaying "01" for all rows, P18; unfilled helper columns, P5). Finally, some workflows reflected static or unstable design choices, such as relying on fixed multipliers for forecasting (P9) or attempting unsupported sorts on array formulas (P12). In every case, the system's corresponding suggestions resolved the underlying issues, producing correct and complete results when executed in the spreadsheet environment.

6.3 Other factors influencing suggestion ratings.

To understand what else might shape participants' ratings, we analyzed several additional factors. We first examined whether properties of the suggestions themselves influenced participants' evaluations. Specifically, we controlled for suggestion length (word and character counts) and structural complexity (e.g., number of paragraph breaks and bullet points). Across all eight evaluation questions, none of these properties had a significant effect on ratings after Holm correction (all $p > .2$), and including them in the models did not alter the significance or direction of the system effect.

We then analyzed several task and participant-related factors, including the task being performed, tool usage order, the number of distinct Excel features referenced in a suggestion, whether participants chose to test the suggestions during the study, and whether the suggestions, when tested, successfully produced the intended outcome. Among these factors, we found no significant effect of

task or tool usage order on suggestion ratings across any of the evaluation questions ($p > .1$ and $p > .3$, respectively).

However, the number of distinct Excel features referenced in a suggestion showed significant effects on several measures, including how clearly participants understood the suggestion ($F = 11.0$, $p = 0.006$, $\beta = 0.14$), whether the suggestions made them aware of more efficient approaches in Excel ($F = 8.7$, $p = 0.019$, $\beta = 0.14$), and whether the suggestion included enough context ($F = 9.9$, $p = 0.011$, $\beta = 0.14$). These results show that suggestions referencing a wider range of built-in Excel features were perceived as clearer, more educational, and better contextualized. Participants who tested suggestions, or for whom the suggestions successfully produced the correct outcome, also gave significantly higher ratings of relevance ($F = 6.5$, $p = 0.049$ and $F = 8.3$, $p = 0.024$, respectively) and were more willing to receive similar suggestions in the future ($F = 6.8$, $p = 0.042$ and $F = 9.0$, $p = 0.017$).

6.4 Revisiting Our Goals

6.4.1 G1. Translate visual representations into semantically rich representations without specialized instrumentation. Rather than relying on instrumentation or logs, InvisibleMentor interprets workflows directly from visual evidence. This grounding in actual behavior led participants to describe the system's suggestions as specific, trustworthy, and aligned with their needs. P3 said it "felt like the system had been watching me and knew exactly where I messed up." P6 appreciated how it "pointed out my error and mistypes," and P13 valued that it identified "where there may be data breakage and how to solve for them." Participants found that the suggestions reflected not just generic advice, but a structured understanding of what they had done, contributing to higher relevance and usefulness.

While Excel provides built-in mechanisms such as VBA macros to log user actions, these methods require users to enable macros and are limited to capturing interactions within a single spreadsheet. Our goal was not to replace such mechanisms, but to explore an alternative that is more lightweight, setup-free, and broadly applicable. By relying solely on screen recordings, our system avoids instrumentation barriers and can reconstruct workflows that span multiple files and applications. This makes InvisibleMentor particularly useful for generating suggestions or feedback after a task is completed, even when no prior logging or setup was in place.

6.4.2 G2. Provide structured, high-fidelity suggestions grounded in user behavior.

InvisibleMentor's suggestions included step-by-step explanations and examples. Beyond clarity, participants valued that InvisibleMentor provided concrete, example-driven guidance. Several participants (P3-4, P10, P12, P14, P17, P20) mentioned that the tool didn't merely point out issues, but offered detailed next steps. P12 highlighted the system's "step by step examples to walk you through," and P20 similarly appreciated the "step-by-step method and explanation of the formulas." P4 emphasized the importance of "list[ing] out in detail what I need to do," while P17 praised the "clear steps and context."

InvisibleMentor's suggestions aligned with participants' skill levels and data contexts. Participants (P3, P5-6, P10-11, P15, P17) reported

that the suggestions met them at the right level of detail and complexity. P5 explained that the guidance was "shorter and more concise," helping them move faster under time pressure. P6 noted that they could "build upon [the suggestion]," and P17 appreciated that the system adapted to larger datasets and included justification for why one method was superior.

6.4.3 G3. Discover unspoken help opportunities by modeling user behavior directly from observation. Most participants (P1-3, P5-9, P11, P13, P15, P19) attributed InvisibleMentor's low-effort interaction to its ability to infer intent from user actions, which eliminated the need to craft explicit prompts. P3 noted that the tool "felt quicker to prompt based on what I was already doing without having to directly posit a question," emphasizing the reduced need for explicit formulation. P1 appreciated that InvisibleMentor "edited [suggestions] as I was attempting it," highlighting its ability to offer in-the-moment guidance without user intervention. P7 further remarked that InvisibleMentor "interpreted your actions so it knew where to prompt you... and highlighted observing actions, which is something I have not seen in other AIs," while contrasting this with the baseline, which "was just asking raw questions based on what you thought." Furthermore, P8 described the baseline as "more like a how-to-do guide" that required users "to be more specific to use [it] effectively."

Behavior-based analysis revealed overlooked opportunities. Participants also noted that InvisibleMentor identified workflow improvements they might not have noticed on their own. For example, P2 remarked that the tool highlighted things "that I did manually which can break or mess up the data." P19 described how InvisibleMentor "offered better solutions for the future," and P13 shared that the system "highlighted other ways to accomplish the task... that I did not think of."

6.4.4 G4. Deliver guidance after task completion to help users notice more efficient alternatives.

Participants appreciated post-task suggestions for helping them recognize where their workflows could be improved. Many participants (P3, P5, P9-P10, P13, P15, P17-P19) appreciated that InvisibleMentor delivered suggestions after they had completed the task, enabling them to identify where their workflows could be improved and to notice more efficient alternatives. P15 noted that the system helped them "understand how to do things better," and P17 likened it to a study guide that "scales with larger datasets," emphasizing its usefulness for recurring tasks. By receiving feedback after completing the task, participants could take time to understand missteps and consider alternative approaches without interrupting their workflow. P2 shared that they would reuse the tool "to pinpoint errors and work from there," indicating a preference for addressing issues after task completion over immediate correction.

A few participants (P2, P11-12, P20) desired real-time suggestions. P11 explicitly asked whether InvisibleMentor could provide help "in the middle of the task," and P12 said "I do find the value of [InvisibleMentor] if I can get more in-real time suggestions."

6.5 Potential to Generalize Beyond Spreadsheets

Many existing assistants depend on application-specific APIs or instrumentation. In contrast, InvisibleMentor operates purely from visual evidence. As a first step in assessing the approach’s potential to apply across applications, we applied the pipeline—without architectural modification—to screen recordings from two distinct domains: computer-aided design (AutoCAD) and weather app development. We made only minor prompt adjustments to generalize terminology (e.g., replacing “cells” with “interface elements”)¹⁰.

We selected both videos using a simple YouTube keyword search (e.g., “AutoCAD practice drawing”, “programming with me”), choosing the top-ranked screen recordings longer than 15 minutes to avoid cherry-picking particular workflows¹¹. Although the recordings are not formal tutorials, their creators are experienced users who typically demonstrate practiced workflows. As a result, the videos contain relatively few inefficient actions, giving the system limited opportunities to identify potential improvements. This makes them conservative test cases for examining how the pipeline behaves in unfamiliar domains.

AutoCAD. In the AutoCAD video, InvisibleMentor identified several opportunities to improve repetitive geometric-editing actions. For example, the user created a rectangular plate, rounded its corners with the Fillet tool, and then manually repositioned the shape. InvisibleMentor recommended drawing the rounded rectangle directly using AutoCAD’s built-in Rectangle-with-Fillet option, reducing a three-step sequence to a single operation¹². The system also observed the user repeatedly adding diameter and radius annotations and manually editing each label to include quantity prefixes (e.g., “2×Ø5.6”, “4×R1.0”). It recommended using Dimension Style Manager or Field attributes to apply such prefixes automatically.

A third recommendation flagged repeated layer changes and suggested using Quick Select to batch-select matching objects before applying color and transparency changes. Two of the three recommendations were directly actionable, while one was less relevant to the designer’s immediate goals. We did not observe false-positive recommendations in this example. Together, these outputs show that the pipeline can detect repetition and inefficient multi-step operations even in complex, canvas-based graphical interfaces.

Weather App Development. In the weather app development video, the developer repeatedly checked weather results in two separate interfaces: their custom app and the OpenWeatherMap website. InvisibleMentor identified this redundancy and suggested relying solely on the app’s built-in API integration once configured.

The system also observed the user copying API credentials through Notepad++ before pasting them into VS Code. It recommended storing the key in a local configuration file and opening it in a split view next to the source code, reducing context switching and keeping relevant information in one place.

Finally, when the developer updated UI elements across multiple parts of the code by manually setting animation classes and visibility of several divs, InvisibleMentor recommended encapsulating these repeated edits into a single function (e.g., `updateWeatherUI(state)`).

All three recommendations were judged reasonable, with one particularly strong workflow improvement.

Summary. Across both examples, InvisibleMentor produced three recommendations that were largely relevant and actionable within their respective workflows. These cases demonstrate how the same pipeline can operate outside spreadsheets with only minor prompt adjustments. Because the examples are drawn from publicly available creator-produced videos, they do not reflect the breadth of naturalistic workflows. Our observations should therefore be interpreted as early evidence of feasibility rather than broad cross-domain generality. A fuller assessment of applicability will require future studies in more varied domains and work settings.

7 Limitations and Future Work

7.1 Limitations

Our findings should be interpreted in light of several limitations. First, our evaluation focused on a set of spreadsheet tasks that were intentionally chosen to be common and well-scoped (e.g., sorting, filtering, and simple formula edits). While these tasks represent realistic use cases, they may not reflect the full diversity of spreadsheet workflows. For example, tasks involving complex macros, cross-sheet dependencies, or domain-specific logic (e.g., financial modeling) may pose challenges for the current system’s suggestion quality and reliability.

Second, InvisibleMentor relies on accurate recognition of user actions from screen recordings. Although our evaluation showed high accuracy for the chosen task types, some actions are harder to detect or interpret. For example, bolding one cell in a large selection or brief font-color changes can be visually subtle. As shown in Section 4.2, short and transient formatting operations are especially prone to being skipped by the VLM’s frame selection mechanism.

In our study tasks, these missed formatting actions did not materially affect the recommendations because the system’s suggestion module operates mainly over functional actions, which were consistently detected. However, formatting plays a more central role in presentation-oriented workflows such as slide authoring, where missed formatting changes may limit the system’s ability to propose improvements. Future work should evaluate the system on tasks where formatting is integral to users’ goals and explore methods for more robust detection of fine-grained visual changes.

Third, our study participants were a mix of novice and spreadsheet-proficient users. While most had prior experience with Excel and AI tools, their behaviors may differ from those of domain experts (e.g., accountants, analysts) or novices. Additionally, the study was conducted in a controlled lab setting. In real-world workflows, time pressure, multitasking, or incomplete task context may affect how users perceive or act on system suggestions.

Fourth, our evaluation does not include a within-paradigm baseline that provides post-task recommendations without visual grounding. Such a comparison could help isolate the specific contribution of the VLM component. We leave this to future work.

Finally, while participants generally reacted positively to the idea of AI-generated suggestions, their actual reliance on them was often opportunistic and selective. Some participants ignored suggestions they found too obvious or irrelevant, while others expressed

¹⁰Updated prompts for both examples are included in the supplemental materials.

¹¹URLs to the videos and updated prompts are provided in the supplemental materials to support reproducibility.

¹²See supplemental material for full recommendations.

hesitation to fully trust AI recommendations. These nuanced behaviors point to the importance of ongoing refinement in timing, relevance, and trust calibration for such systems.

7.2 Future Work

Enhancing system reliability and user-directed assistance. A broader direction for future work is to increase the reliability of the system's inferences while giving users more control over how assistance is provided. Currently, InvisibleMentor offers no mechanism for revising or verifying misinterpreted actions, which can potentially propagate errors into the generated suggestions. Future iterations could incorporate self-correction strategies that detect low-confidence predictions, re-query recent frames, or allow users to backtrack and adjust a misunderstanding. In this context, lightweight replay of the system's inferred action sequence could help users verify what the system believed they did and correct any inaccuracies before suggestions are generated. Such feedback loops should be designed carefully to avoid overwhelming users, aligning with longstanding mixed-initiative principles that emphasize calibrating system initiative with user control [3, 25].

A complementary direction is to pair reflective suggestions with light-weight execution support. While our system focuses on interpreting user behavior and generating structured multi-step recommendations, future systems could help users act on these suggestions more directly, like our baseline, Excel Copilot. But, beyond simple click-to-apply operations, an assistant could consider providing step-by-step demonstrations or short replays of the recommended procedure, enabling users to follow along until they feel confident executing the workflow independently. Achieving such capabilities would require access to stable application-side action primitives. Future systems could incorporate an additional action layer, such as those provided by the Model Context Protocol (MCP), to bridge recommendations with controllable application operations while preserving user oversight.

8 Conclusion

InvisibleMentor turns passive screen recordings into active mentorship. By analyzing visual evidence rather than system logs, it identifies inefficient workflows and suggests better alternatives without requiring user prompts. Our evaluation confirms that users find these suggestions clear, accurate, and easy to apply. InvisibleMentor demonstrates that AI can do more than just answer questions or automate away tedious tasks; it can teach us to master the tools we thought we knew.

Acknowledgments

References

- [1] David Akers, Matthew Simpson, Robin Jeffries, and Terry Winograd. 2009. Undo and erase events as indicators of usability problems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 659–668.
- [2] Mohammad Alahmadi, Abdulkarim Malkadi, and Sonia Haiduc. 2020. UI Screens Identification and Extraction from Mobile Programming Screencasts. In *Proceedings of the 28th International Conference on Program Comprehension*. ACM, 319–330.
- [3] J.E. Allen, C.I. Guinn, and E. Horvitz. 1999. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications* 14 (1999), 14–23.
- [4] Sara M. Alspaugh. 2017. *Understanding Data Analysis Activity via Log Analysis*. Ph.D. Dissertation. University of California, Berkeley.
- [5] Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. ScreenAI: a vision-language model for UI and infographics understanding. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*. Article 339, 11 pages.
- [6] Carlos Bernal-Cárdenas, Nathan Cooper, Madeleine Havranek, Kevin Moran, Oscar Chaparro, Denys Poshyvanyk, and Andrian Marcus. 2023. Translating Video Recordings of Complex Mobile App UI Gestures into Replayable Scenarios. *IEEE Transactions on Software Engineering* 49 (2023), 1782–1803.
- [7] Ann Blandford, Dominic Furniss, and Stephann Makri. 2016. *Qualitative HCI research: Going behind the scenes*. Morgan & Claypool Publishers.
- [8] David A. Bradbard, Charles Alvis, and Richard Morris. 2014. Spreadsheet usage by management accountants: An exploratory study. *Journal of Accounting Education* (2014), 24–30.
- [9] Tyson Bulmer, Lloyd Montgomery, and Daniela Damian. 2018. Predicting developers' IDE commands with machine learning. In *Proceedings of the 15th International Conference on Mining Software Repositories*. ACM, 82–85.
- [10] John M. Carroll and Mary Beth Rosson. 1987. *Paradox of the active user*. MIT Press, Cambridge, MA, USA, 80–111.
- [11] George Chalhoub and Advait Sarkar. 2022. "It's Freedom to Put Things Where My Mind Wants": Understanding and Improving the User Experience of Structuring Data in Spreadsheets. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, Article 585, 24 pages.
- [12] Sibe Chen, Yeye He, Weiwei Cui, Ju Fan, Song Ge, Haidong Zhang, Dongmei Zhang, and Surajit Chaudhuri. 2024. Auto-Formula: Recommend Formulas in Spreadsheets using Contrastive Learning for Table Representations. *Proceedings of the ACM on Management of Data*, Article 122 (2024), 27 pages.
- [13] Yanting Chen, Yi Ren, Xiaoting Qin, Jue Zhang, Kehong Yuan, Lu Han, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2024. Sharingan: Extract User Action Sequence from Desktop Recordings.
- [14] Parmit K. Chilana, Amy J. Ko, and Jacob O. Wobbrock. 2012. LemonAid: selection-based crowdsourced contextual help for web applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1549–1558.
- [15] Allen Cypher. 1993. *Eager: programming repetitive tasks by demonstration*. MIT Press, Cambridge, MA, USA, 205–217.
- [16] Robert DeLine, Amir Khella, Mary Czerwinski, and George Robertson. 2005. Towards understanding programs through wear-based filtering. In *Proceedings of the 2005 ACM Symposium on Software Visualization*. ACM, 183–192.
- [17] Travis Faas, Lynn Dombrowski, Alyson Young, and Andrew D. Miller. 2018. Watch Me Code: Programming Mentorship Communities on Twitch.tv. *Proceedings of the ACM on Human-Computer Interaction*, Article 50 (2018), 18 pages.
- [18] Leah Findlater and Joanna McGrenere. 2004. A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 89–96.
- [19] Adam Fourney, Richard Mann, and Michael Terry. 2011. Query-feature graphs: bridging user vocabulary and system functionality. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, 207–216.
- [20] C. Ailie Fraser, Mira Dontcheva, Holger Winnemöller, Sheryl Ehrlich, and Scott Klemmer. 2016. DiscoverySpace: Suggesting Actions in Complex Software. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. ACM, 1221–1232.
- [21] Tovi Grossman and George Fitzmaurice. 2010. ToolClips: an investigation of contextual video assistance for functionality understanding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1515–1524.
- [22] Sumit Gulwani, William R. Harris, and Rishabh Singh. 2012. Spreadsheet data manipulation using examples. *Commun. ACM* (2012), 97–105.
- [23] Björn Hartmann, Daniel MacDougall, Joel Brandt, and Scott R. Klemmer. 2010. What would other programmers do: suggesting solutions to error messages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1019–1028.
- [24] Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* (1979), 65–70.
- [25] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 159–166.
- [26] Forrest Huang, Gang Li, Tao Li, and Yang Li. 2024. Automatic Macro Mining from Interaction Traces at Scale. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, Article 1038, 16 pages.
- [27] Yue Jiang, Eldon Schoop, Amanda Swearngin, and Jeffrey Nichols. 2025. ILuvUI: Instruction-tuned LangUAge-Vision modeling of UIs from Machine Conversations. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*. ACM, 861–877.
- [28] Yiqiao Jin, Stefano Petrangeli, Yu Shen, and Gang Wu. 2025. ScreenLLM: Stateful Screen Schema for Efficient Action Understanding and Prediction. In *Companion Proceedings of the ACM on Web Conference 2025*. ACM, 2008–2013.
- [29] Anjali Khurana, Xiaotian Su, April Yi Wang, and Parmit K Chilana. 2025. Do It For Me vs. Do It With Me: Investigating User Perceptions of Different Paradigms of

- Automation in Copilots for Feature-Rich Software. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. ACM, Article 880, 18 pages.
- [30] Anjali Khurana, Hariharan Subramonyam, and Parmit K Chilana. 2024. Why and When LLM-Based Assistants Can Go Wrong: Investigating the Effectiveness of Prompt-Based Interactions for Software Help-Seeking. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. ACM, 288–303.
- [31] Andrea Kohlhasse, Michael Kohlhasse, and Ana Guseva. 2015. Context in Spreadsheet Comprehension. In *SEMS@ ICSE*. 21–27.
- [32] Benjamin Lafreniere, Andrea Bunt, Matthew Lount, Filip Krynicki, and Michael A. Terry. 2011. AdaptableGIMP: designing a socially-adaptable interface. In *Proceedings of the 24th Annual ACM Symposium Adjunct on User Interface Software and Technology*. ACM, 89–90.
- [33] Benjamin Lafreniere, Parmit K. Chilana, Adam Fourney, and Michael A. Terry. 2015. These Aren't the Commands You're Looking For: Addressing False Feedback in Feature-Rich Software. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 619–628.
- [34] Jenny T. Liang, Aayush Kumar, Yasharth Bajpai, Sumit Gulwani, Vu Le, Chris Parnin, Arjun Radhakrishna, Ashish Tiwari, Emerson Murphy-Hill, and Gustavo Soares. 2025. TableTalk: Scaffolding Spreadsheet Development with a Language Agent. *ACM Transactions on Computer-Human Interaction* 32, Article 65 (2025), 49 pages.
- [35] Wendy E. Mackay. 1990. Patterns of sharing customizable software. In *Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work*. ACM.
- [36] Abdulkarim Malkadi, Ahmad Tayeb, and Sonia Haiduc. 2023. Improving Code Extraction from Coding Screencasts Using a Code-Aware Encoder-Decoder Model. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 1492–1504.
- [37] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2011. Ambient help. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2751–2760.
- [38] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2013. Patina: dynamic heatmaps for visualizing application usage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3227–3236.
- [39] Justin Matejka, Wei Li, Tovi Grossman, and George Fitzmaurice. 2009. CommunityCommands: command recommendations for software applications. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*. ACM, 193–202.
- [40] Emerson Murphy-Hill, Rahul Jiresal, and Gail C. Murphy. 2012. Improving software developers' fluency by recommending development environment commands. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. ACM, Article 42, 11 pages.
- [41] Emerson Murphy-Hill, Da Young Lee, Gail C Murphy, and Joanna McGrenere. 2015. How do users discover new tools in software development and beyond? *Computer Supported Cooperative Work (CSCW)* 24, 5 (2015), 389–422.
- [42] Aadhavan M. Nambhi, Bhanu Prakash Reddy, Aarsh Prakash Agarwal, Gaurav Verma, Harvineet Singh, and Ifitikhar Ahamath Burhanuddin. 2019. Stuck? No worries! Task-aware Command Recommendation and Proactive Help for Analysts. In *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization*. ACM, 271–275.
- [43] Jakob Nielsen. 1994. *Usability Engineering*. Morgan Kaufmann Publishers Inc.
- [44] Chris Parnin and Robert DeLine. 2010. Evaluating cues for resuming interrupted programming tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 93–102.
- [45] Vidya Ramesh, Charlie Hsu, Maneesh Agrawala, and Björn Hartmann. 2011. ShowMeHow: translating user interface instructions between applications. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, 127–134.
- [46] Franklin E. Satterthwaite. 1946. An approximate distribution of estimates of variance components. *Biometrics bulletin* 2, 6 (1946), 110–114.
- [47] Rishabh Singh and Sumit Gulwani. 2016. Transforming spreadsheet data types using examples. *POPL '16: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Language* (2016), 343–356.
- [48] Ananya Singha, Bhavya Chopra, Anirudh Khatri, Sumit Gulwani, Austin Henley, Vu Le, Chris Parnin, Mukul Singh, and Gust Verbruggen. 2024. Semantically Aligned Question and Code Generation for Automated Insight Generation. In *Proceedings of the 1st International Workshop on Large Language Models for Code*. ACM, 127–134.
- [49] Sruti Srinivasa Ragavan, Advait Sarkar, and Andrew D Gordon. 2021. Spreadsheet Comprehension: Guesswork, Giving Up and Going Back to the Author. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Article 181, 21 pages.
- [50] Michael B. Twidale. 2005. Over the shoulder learning: supporting brief informal learning. *Computer Supported Cooperative Work (CSCW)* 14, 6 (2005), 505–547.
- [51] Xu Wang, Benjamin Lafreniere, and Tovi Grossman. 2018. Leveraging Community-Generated Videos and Command Logs to Classify and Recommend Software Workflows. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 1–13.
- [52] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in Statistics: Methodology and Distribution*. Springer, 196–202.
- [53] Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Liu Jianfeng, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024. MobileVLM: A Vision-Language Model for Better Intra- and Inter-UI Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. ACL, 10231–10251.
- [54] Qing (Nancy) Xia, Advait Sarkar, Duncan P. Brumby, and Anna L. Cox. 2025. 'How Do You Know That Stuff?': Barriers to Expertise Sharing Among Spreadsheet Users. *Proc. ACM Hum.-Comput. Interact.* 9, 7, Article CSCW230 (2025), 26 pages.
- [55] Pingjing Yang and Ti-Chung Cheng. 2020. Understanding Data Analysis Workflows on Spreadsheets: Roadblocks and Opportunities. In *Proceedings of Workshop on Human-In-the-Loop Data Analytics (HILDA'20)*.
- [56] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Article 437, 21 pages.
- [57] Dehai Zhao, Zhenchang Xing, Chunyang Chen, Xin Xia, and Guoqiang Li. 2019. ActionNet: Vision-Based Workflow Action Recognition From Programming Screencasts. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 350–361.
- [58] Dehai Zhao, Zhenchang Xing, Xin Xia, Deheng Ye, Xiwei Xu, and Liming Zhu. 2023. SeeHow: Workflow Extraction from Programming Screencasts through Action-Aware Video Analytics. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 1946–1957.

A Prompt Templates

This appendix provides the prompt templates used in each phase of InvisibleMentor's architecture. The prompts were constructed to guide the vision-language and language models in extracting meaningful task representations and generating grounded, actionable recommendations. These prompts were issued via the OpenAI API during system execution.

A.1 VLM Prompt for Task Representation Extraction

We use the GPT-4.1 vision-language model to extract structured user actions and spreadsheet content from screen recordings. The prompt is shown below:

You are an assistant for workflow analysis. Given a sequence of frames from a task video and a list of prior identified actions, analyze the frames and identify any new user actions that are not already described in the prior actions. If you find new actions, add them to the new_actions. If no new actions are identified, return empty new_actions.

You will be provided with a batch of frames and a list of prior actions. For each action, you need to identify all of the details, such as the formatting, "cell content", "formula", and the specific action. Notice that some actions may have different outcomes based on the content of the sheet. For example, if the user switches to a different sheet, it may create a new sheet in addition to switching to it.

For some actions not shown on the frames, you need to predict them by comparing the difference between the frames. For example, whether the styles, formatting, or content of the cells have changed. If the user has changed the content of a cell, you need to speculate the possible actions that may lead to the changes and add the actions to the new_actions.

If the content of the sheet has changed (not cell formats), you need to record the entire sheet in Markdown format. Remember to include the workbook name and sheet name.

Your response should be a JSON object with the following structure:

```
{
  "new_action_detected": true/false,
  "new_actions": [
    "action1",
    "action2",
    ...
  ],
  "sheet_changes": true/false,
  "sheet_details": "use Markdown format to record the entire sheet"
}
```

6. Use proper formatting: backticks (`) around Excel functions, formulas, keyboard shortcuts, and feature names, and triple backticks (```) for multi-line formulas or step-by-step code examples
7. Create plausible placeholders for unclear data references

Output JSON format:

```
{
  "Workflows": [
    {
      "ActionList": ["Action 1", "Action 2"],
      "Optimal": true/false,
      "Reason": "Brief explanation",
      "Suggestion": "Step-by-step actionable solution"
    }
  ]
}
```

A.2 LLM Prompt for Recommendation Generation

To generate recommendations, we use a textual LLM (OpenAI o3 model) prompted with a structured representation of the user's observed actions and spreadsheet state. The prompt instructs the model to identify suboptimal workflows, explain their inefficiencies, and suggest improved methods using built-in Excel features. The model is asked to provide step-by-step instructions grounded in the user's context, and to rank recommendations by impact.

You are a workflow efficiency expert. Analyze user actions from Excel task videos and identify suboptimal workflows.

Instructions:

1. Group related actions into workflows (steps accomplishing a specific task)
2. For each workflow, set "Optimal" to true/false based on efficiency
3. For suboptimal workflows ("Optimal": false):
 - "ActionList": List actions starting with "It looks like you..."
 - "Reason": Main inefficiency (be specific) starting with "You ..."
 - "Suggestion": Provide ONE actionable solution using Excel features:
 - Give step-by-step instructions with exact Ribbon paths/shortcuts
 - Include detailed examples with realistic sheet/column names
 - Prioritize automation over manual repetition
 - Provide complete formulas with explanations when applicable
 - End with "Benefit:" explaining concrete improvements (time saved, fewer steps, error reduction)
 - Compare before/after: "Original: X steps, Suggested: Y steps"
4. Focus on efficiency and maintainability, not just task completion
5. Only include 3 most impactful suboptimal workflows and rank them by importance

B User Scenario

Mia, an analyst working with sales data, wants to calculate the total revenue for each city. Her goal seems straightforward, but she isn't sure of the best approach. She tries several methods: filtering rows manually, using combinations of SUM() and IF() formulas, hovering over columns to check auto-totals, and eventually constructing a UNIQUE column with corresponding SUMIF() formulas. These attempts span different sheets, ranges, and parameter values, many of which she adjusts or discards along the way.

Later, she wonders if there's a more efficient solution, and considers asking a chatbot. But expressing her workflow turns out to be harder than expected. She vaguely recalls that SUMIF "didn't work" at first, but doesn't remember why. It may have been due to a mismatched range, but she can't verify it. In describing her steps, she omits earlier failed attempts and small but relevant actions like adjusting cell formats or deleting helper columns. Even when she tries to be thorough, she struggles to name certain tools ("that dropdown with formula functions") or describe all the cell styles she modified. Faced with the burden of reconstructing everything in words, she gives up on asking for help.

Instead, she turns to InvisibleMentor, which generates recommendations based on her observed interactions, without requiring her to describe them (G1). In the sidebar, one of the existing five prompt ideas has been repurposed to serve as an entry point for InvisibleMentor, and Mia selects it (Figure 3, ❶). This selection issues a pre-populated request on her behalf, triggering the assistant to display a recommendation grounded in her actual workflow.

Next, InvisibleMentor summarizes Mia's recent activity, such as building a summary table with UNIQUE and SUMIF() formulas, and identifies limitations of this approach (Figure 3, ❷, ❸). By tying its suggestions to specific steps in her process, the system keeps its feedback directly relevant.

To help her improve, InvisibleMentor offers a clear, step-by-step guide for using PivotTables to achieve the same goal more efficiently (Figure 3, ❹) (G2). This recommendation appears immediately after she finishes building her summary table, while the task is still salient and her cognitive load has eased (G4). She can also request alternative suggestions using a dedicated "give me another suggestion" option (Figure 3, ❺).

With minimal effort, Mia produces a more maintainable summary and learns a new feature that improves her workflow. The assistant bridges the gap between what she did and what she could have done better, without requiring her to reconstruct her process or translate it into a prompt (G3).

C Additional Details for Evaluation 1

C.1 Task Overview and Spreadsheet Contents

In Evaluation 1, we used a set of benchmark tasks that span a range of common spreadsheet activities. These tasks were performed in the browser using a standardized Excel file containing two sheets:

- **Budget sheet:** Includes 7 rows (Title, empty row, and 5 categories—Category, Rent, Car, Food, Total) and 5 columns (Category, Jan, Feb, Mar, Apr). Cells were pre-filled with monthly budget data (e.g., Rent = \$1,275).
- **Address History sheet:** Contains 3 rows of addresses with columns for Street Line 1, Line 2, City, State, Zip Code, and Move Date (From / To). Example: '33 Cherry Drive, N/A, San Diego, CA, 92104, June 2021, Present'.

Participants completed the following 14 tasks using the Excel web app:

- (1) **Open file:** Access a spreadsheet via a link in a Gmail message.
- (2) **Add row:** Insert a new row (e.g., a new category) in the Budget sheet.
- (3) **Add values:** Input numerical values (e.g., monthly expenses) into the spreadsheet.
- (4) **Edit formula:** Modify or insert a SUM formula to update a monthly total.
- (5) **Bold:** Apply bold formatting to cell text.
- (6) **Cell fill color:** Use fill color to highlight a cell (e.g., coloring column header).
- (7) **Switch sheet:** Navigate between the 'Budget' and 'Address History' sheets.
- (8) **Resize column:** Adjust the width of a column (e.g., Street Name column).
- (9) **Copy/paste:** Duplicate content within the spreadsheet.
- (10) **Change font color:** Modify the font color of selected cells.
- (11) **New sheet:** Create a new, blank worksheet in the workbook.
- (12) **Rename sheet:** Rename an existing worksheet (e.g., rename 'Sheet1' to 'Accounts').
- (13) **Share link:** Generate and share a link to the workbook via email.
- (14) **New workbook:** Start a new workbook from the Excel start screen.

These tasks were selected to cover a range of editing, formatting, navigation, and collaboration operations commonly encountered in everyday spreadsheet use.

C.2 VLM Processing Time and Efficiency

In addition to measuring action extraction accuracy, we recorded how long the Vision-Language Model (VLM) took to process each benchmark screen recording during Evaluation 1 (Figure C.2). Because participants took varying amounts of time to complete each task, the video durations ranged from 9.8 minutes to 49.5 minutes.

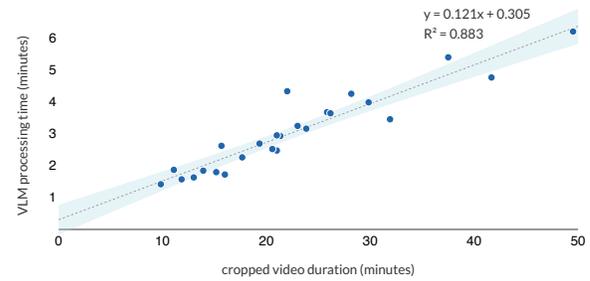


Figure 6: Relationship between video duration and VLM processing time. Each dot represents one screen recording session from Evaluation 1. The x-axis represents the duration of the cropped input video, and the y-axis shows the time taken by the VLM to process that session. A fitted linear regression line indicates a strong positive correlation ($R^2 = 0.883$).

We measured the wall-clock time required for the VLM to process each video, recorded during our technical evaluation under the same experimental setup (see Section 4.1). For clarity, the processing times reported here were measured without parallel processing. Frames were sampled at fixed 5-second intervals and passed to the model for inference. The average processing time was 2.99 minutes ($SD = 1.18$). The shortest video (9.8 minutes) took 1.41 minutes, while the longest (49.5 minutes) took 6.20 minutes. Figure 6 shows a dot plot of video duration versus processing time, with a fitted regression line indicating a strong linear relationship.

D Task Instructions

This appendix provides the full task descriptions shown to participants during Study 2. Tasks were designed to simulate realistic spreadsheet analysis scenarios involving multi-step operations over large, multi-sheet workbooks.

D.1 Task A: Vrinda Store Sales Analysis

Vrinda is an exclusive clothing store offering a wide range of men's and women's wear tailored to various age groups. The goal of this task is to analyze Vrinda's sales data for the year 2022 to gain insight into the customer base and identify opportunities for revenue growth.

Task Description. Using Microsoft Excel and the provided dataset, please complete the following analysis tasks:

- (1) Create a single chart that compares total sales (Amount) and the number of orders (Qty) by month for 2022.
- (2) Identify which month had the highest total sales and the most orders.
- (3) Analyze whether women or men made more purchases in 2022.
- (4) Use the 2022 sales trend to forecast monthly sales for 2023.

Please spend about 8 minutes on each question and do your best within that time. Perfection is not expected. You will be notified when one minute remains, and when it is time to move on to the next task. There are no right or wrong answers—we are simply interested in seeing your natural approach.

D.2 Task B: Pizza Sales Analysis

In this task, you take on the role of a BI consultant hired by Plato's Pizza, a Greek-inspired restaurant in New Jersey. Your objective is to analyze the restaurant's performance using historical order data and identify operational insights.

Task Description. Using Microsoft Excel and the provided dataset, please complete the following analysis tasks:

- (1) Determine which days of the week and which times of day tend to be the busiest.
- (2) Estimate how many pizzas are made during peak periods.
- (3) Identify the best-selling and worst-selling pizzas.

- (4) Calculate the average order value.

As before, spend approximately 8 minutes on each question. You will be notified near the end of each segment. The goal is not perfection—we are simply interested in understanding how you approach realistic Excel tasks.

Instructions for Both Tasks. When you are ready to begin, please inform the experimenter. You may use any standard Excel features during the task. The interface will also include a side panel that displays system-generated recommendations (depending on condition). You are encouraged to explore and apply these recommendations during the task.