# General Accessibility Model: Understanding Accessibility Needs from Computer Use

ANONYMOUS AUTHOR(S)

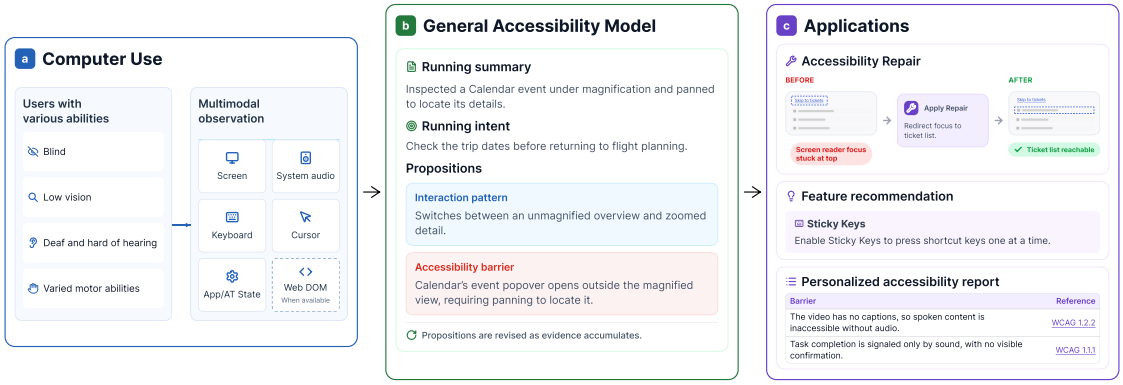



Fig. 1. General Accessibility Model (GAM) infers accessibility needs from observing computer use. (a) Multimodal traces provide evidence of interaction across access methods. (b) GAM maintains a running summary, task intent, and propositions about interaction patterns and accessibility barriers. (c) GAMUT applications use this representation for accessibility repairs, feature recommendations, and personalized accessibility reports.

Accessibility research has long envisioned systems that adapt to users' abilities and contexts. Yet current ability-based systems lack reliable ways to learn users' accessibility needs in situ, leaving many individualized and context-specific barriers unaddressed. We present the **General Accessibility Model (GAM)**, which learns accessibility needs by observing computer use. From screen recordings, input traces, and interface context, GAM generates propositions describing a user's interaction strategies and in-situ accessibility barriers. This understanding enables accessibility repairs, feature recommendations, and personalized accessibility reports. We evaluated GAM on a self-collected dataset of barrier scenarios and in a study with ten participants who were blind, had low vision, or had motor impairments. GAM recovered most barriers labeled major, and most generated repairs resolved their targeted barrier in post-hoc testing. Through GAM, we envision user interaction becoming a continuously updated source of knowledge for personal accessibility support.

CCS Concepts: • **Human-centered computing** → **Accessibility systems and tools**.

Additional Key Words and Phrases: accessibility, assistive technologies, user modeling, interface repair, web accessibility, large multimodal models

## 1 Introduction

Many people encounter barriers to computing when the interface does not account for their abilities or situational context. For decades, accessibility research has worked to identify these *accessibility barriers* and develop technologies to address them. A significant body of work has concentrated on detecting barriers within an interface. For example, using a set of general accessibility guidelines (e.g., the Web Content Accessibility Guidelines (WCAG) [52]), researchers have shown that a staggering number of the top million websites (95.9%) contain accessibility barriers or violations of

these guidelines [50]. These guidelines are invaluable to understanding the overall accessibility of computing, informing the design of assistive technologies (AT), and providing actionable insights to developers. However, conformance to these guidelines does not guarantee that an interface is accessible to a particular user, which requires understanding the barriers they encounter during actual use [34]. As a result, more recent thinking in accessibility has advocated for understanding barriers and opportunities that arise from the perspective of a person's unique abilities and context [51]. Examples include interfaces that adapt themselves to a user's abilities [12], activities [20], and context [21]. Compared with solutions based on general guidelines, these designs and interventions can target the barriers experienced by a particular user and even identify accessibility improvements that the user may not have known could benefit them [1, 54].

Building the understanding needed to support personalized accessibility is difficult. Prior approaches have often relied on heuristics tied to predefined assumptions about user needs, e.g., thresholding sensor data to infer motor actions [20]. Because these heuristics can only recognize a small set of barriers, moving beyond them requires a more holistic understanding of how a particular person interacts with an interface, including their task, the AT they may already use, input methods, and observed actions. Together, these signals can reveal barriers that are difficult to infer from the interface alone [3, 19, 22, 47]. For example, a page may display a confirmation message after form submission, but a user's magnified viewport may place it outside their view; repeatedly submitting the form can then reveal the practical consequence of that barrier.

Transforming this breadth of interaction evidence into a general representation that can support accessibility interventions is challenging. To this end, recent approaches based on large language models (LLMs), such as General User Models (GUMs), offer a promising starting point: they convert unstructured observations of computer use into natural language propositions about a person's behavior and preferences, revising those propositions as evidence accumulates [41]. Yet, this representation remains incomplete for understanding experienced accessibility barriers because it must also account for the gap between what the model can observe and what the person can perceive or operate. For example, an unmagnified screenshot may show content outside the person's magnified viewport, while a Document Object Model (DOM) snapshot may contain text that their screen reader has not announced. In this case, inferring the barrier requires interpreting interface information alongside evidence of what was actually available to the person, what they attempted, and what feedback they received.

In this paper, we present the **General Accessibility Model (GAM)**, an architecture for inferring accessibility needs from computer use (Figure 1). GAM uses a multimodal LLM to interpret interaction data, including screen recordings, user inputs, and interface context, and to infer and update natural language propositions about recurring interaction patterns and accessibility barriers. Here, "general" refers to a shared approach for representing needs across computer use tasks and people with different access needs. For example, for a participant with a motor impairment searching for accommodations using a budget slider, GAM identified the recurring *interaction pattern* ``When adjusting range sliders, the person experiences extended adjustment loops … with overshoots.'' The same approach can also capture a specific *accessibility barrier* from multi-channel interaction evidence. For a participant with low vision using macOS Calendar to inspect an event, GAM surfaced ``Calendar opens the event popover … outside the magnified view, requiring panning to locate it.'' Figure 1 illustrates how GAM generates and updates these propositions from observed computer use.

With this representation, we introduce **GAMUT** (GAM UTilities), three prototype applications that illustrate how GAM can inform personalized accessibility support (Figure 1). First, *accessibility repair* uses the interaction patterns and barriers identified by GAM to generate targeted, just-in-time, interface modifications, e.g., adding numeric inputs

alongside difficult-to-use sliders or fixing a keyboard trap for a screen reader user. Second, *feature recommendations* connect a person's observed difficulties with existing accessibility features and interaction techniques that may help them. Finally, *personalized accessibility reports* summarize the person's access strategies, encountered barriers, and supporting evidence to explain difficulties in the context of their computer use. Together, this suite of applications shows how the same representation can support personalized interventions both within and beyond the interface.

We evaluate GAM using two datasets we curated. **GAM-Eval-RB** (Reproduced Barriers) contains 62 recordings in which we reproduced audited accessibility barriers on real interfaces. **GAM-Eval-US** (User Study) contains recordings from ten participants who are blind, have low vision, or have motor impairments. Across both datasets, we assess proposition accuracy, recall of documented barriers, and repair effectiveness. Of 222 reviewed unmatched barrier and interaction pattern propositions, 215 (96.8%) were judged accurate. Under full observation, GAM recovered 95% of documented barriers in GAM-Eval-RB and 61% in GAM-Eval-US. Researcher testing on live websites with the relevant AT judged 103 of 132 generated repairs (78.0%) to resolve their targeted barrier. Participants rated repairs as helpful in 26 of 30 responses, including 13 ratings of live-generated repairs. We also compare GAM under full observation (standard GAM pipeline) with a static-input configuration that omits video, audio, and input traces. On GAM-Eval-US, full observation recovered 61% of documented barriers, compared with 32% under static input, with the largest gain among participants with low vision. These results suggest our interaction-based approach can provide a more complete understanding of accessibility needs in computer use.

Overall, this paper contributes:

- The **General Accessibility Model (GAM)**, an architecture that infers accessibility needs from computer use and represents them as natural language propositions about interaction patterns and accessibility barriers.
- A demonstration of **GAMUT**, prototype applications built on GAM's shared representation: accessibility repair, feature recommendations, and personalized accessibility reports.
- An evaluation using recordings of reproduced accessibility barriers (**GAM-Eval-RB**) and recordings from a user study (**GAM-Eval-US**) that assesses proposition accuracy, recall of documented barriers, differences between input configurations, and the effectiveness and perceived usefulness of generated repairs.

## 2 Related Work

To contextualize GAM, we draw on three bodies of prior work: (i) accessibility testing and evaluation, (ii) understanding individual accessibility needs, and (iii) systems that adapt to individual needs.

### 2.1 Identifying Accessibility Barriers

Our work aims to identify accessibility barriers from observations of how people actually use computing systems. Related to this goal, a large body of research has focused on identifying accessibility barriers using a set of accessibility standards such as WCAG [52]. Automated checkers such as axe and WAVE are examples of products that inspect web interfaces for violations using a combination of detection scripts run on the code and rendered page [8, 48]. One body of research has sought to characterize accessibility barriers at scale across web and mobile ecosystems using these types of automated checks [11, 36, 50]. For example, WebAIM's 2026 analysis found detectable WCAG failures on 95.9% of one million home pages [50], and Fok et al. analyzed accessibility failures in 312 Android apps and found that many of them persisted across releases [11].

A drawback of these automated interface scanners is that they focus on accessibility barriers that can only be determined from the interface alone, leading them to underestimate errors that a user might actually experience. A benchmark of six evaluation tools found that none covered more than half of the WCAG 2.0 success criteria [46], and even when experts are asked to apply these guidelines, researchers found that experts interpreted and applied them inconsistently [6, 7]. Several later studies have confirmed that actual problems encountered during use are often not fully covered by conformance criteria [17, 33, 34]. Finally, recent systems such as GenA11y and AccessGuru use language models to reason about interface semantics that are difficult to capture with fixed rules [10, 14]. While these approaches broaden the range of barriers that can be inferred from an interface, they still cannot determine which barriers a particular person would actually encounter during use.

To this end, other approaches to accessibility evaluation have incorporated evidence from automatic operation of the interface. For example, several research systems have used accessibility services to execute interactions on an app rather than relying on interface inspection alone [37–39]. A11yPuppetry is an example of a system that records touch interactions and replays the same flow through assistive technology [38]. Similarly, AXNav translates natural-language test instructions into actions performed through VoiceOver and other iOS accessibility features [43]. Other systems [59, 60] gather evidence of potential interaction barriers through automated traversal of a screen with AT. These systems can reveal barriers that emerge during task execution, but the interaction they observe is still not grounded in the experiences of the end user.

Prior research suggests that observations of people's own experiences are necessary for understanding the computing barriers they face. For example, several studies of blind and low-vision users show that accessibility conditions change how people navigate and how interaction unfolds over time [3, 5, 22, 27]. Several datasets have aimed to capture real computer-use traces collected from people using assistive technology [2, 27]. These datasets capture how people use computers, but not what those interactions reveal about their access needs or how those needs might be addressed. In this paper, we present GAM as an architecture for generating actionable accessibility propositions from an end-user's own computer usage.

### 2.2 Modeling Individual Access Needs

GAM aims to turn observations of computer use into a broader understanding of a person's access needs. Prior accessibility research has approached this problem by modeling individual abilities and interaction patterns. For example, Ability-Based Design argues that systems should accommodate people's abilities rather than require people to adapt to the system [51]. Systems such as SUPPLE and SUPPLE++ applied this idea by using models of user ability to generate interfaces, including models derived from measured motor performance [12, 13]. Finally, shared user models further allowed information about an individual's interaction abilities to guide adaptation across applications [29].

Rather than requiring access needs to be specified directly, prior work has inferred them from interaction behavior [18]. Low-level input traces such as typing and touchscreen interactions have been used to identify suitable accessibility configurations [32, 45]. Other systems draw on broader patterns of use to guide adaptation, including navigation strategies, ordinary device behavior, and observations used to update an ability profile [21, 24, 54]. These approaches infer access needs from behavior, but the properties they seek to recognize are generally defined in advance.

Recent LLM-based approaches provide a more open-ended representation by deriving natural-language propositions directly from observations of computer use [41]. GUM generates and revises these propositions as evidence accumulates, allowing recurring behaviors to emerge from use rather than being defined in advance. Yet, to be useful for accessibility applications, these observations must be interpreted in the context of what information and controls

were actually available to the person through their access methods. GAM introduces an architecture that combines observations of computer use with evidence of the person's access context to infer recurring interaction patterns and experienced accessibility barriers.

### 2.3 Personalized Accessibility Support

Understanding a person's access needs can inform changes to interfaces they already use without requiring developer intervention [9]. Prior research has explored approaches to overcome accessibility barriers using externally authored scripts, accessibility metadata, annotations, or mediation layers to alter how an existing interface is presented or operated [4, 44, 57, 58].

Other work has automated the modification itself. Opportunistic Accessibility Improvement applies magnification while checking for adverse changes to the page, while Screen Recognition reconstructs missing accessibility metadata from rendered pixels [1, 56]. Generative systems extend this capability by filtering content for a user's task, restructuring pages for screen reader navigation, or repairing detected accessibility violations [10, 26, 28, 30, 55]. These systems automate how a known accessibility need is addressed rather than infer broader access needs from a person's use.

Access needs can support more than interface repair. Usage signals can inform recommendations for existing accessibility features [54], while accessibility findings can be organized into reports [42]. Rather than infer these needs separately for each application, GAM maintains a shared account of recurring interaction patterns and experienced barriers. GAMUT reuses this account across interface repair, accessibility recommendations, and personalized reporting.

## 3 General Accessibility Model (GAM)

GAM uses a multimodal LLM to construct and revise a representation of a user's accessibility needs from computer use. The representation consists of *interaction patterns*, which describe recurring ways a person interacts, and *accessibility barriers*, which describe accessibility barriers situated in particular interfaces. Following GUM [41], these propositions use natural language, are grounded in observations, and are revised as evidence accumulates. Here, "general" refers to using this shared approach across tasks, applications, and people with different access needs. The same representation can describe a screen reader navigation strategy on websites, or a barrier the user encountered using a magnifier in a native application, or recurring typing errors across applications.

GAM builds this account from ongoing observation. It splits continuous interaction traces into segments, prepares the available interaction data, interprets each segment alongside accumulated understanding, and updates the current understanding. Each update forms the basis for subsequent inference and makes the representation available to GAMUT applications.

### 3.1 Representing Accessibility Needs

*3.1.1 Interaction pattern propositions.* Interaction pattern propositions describe recurring strategies or preferences in AT use, interface navigation, workarounds, and input and control. For example, GAM described a magnifier user's interaction pattern: ``The person operates macOS Zoom dynamically, surveying full page layouts unmagnified before zooming in to inspect specific item details, sizing, and cart content.'' This describes how the person balances overview and detail, beyond identifying that magnification is active. In our prototype, interaction patterns remain available across tasks and applications.

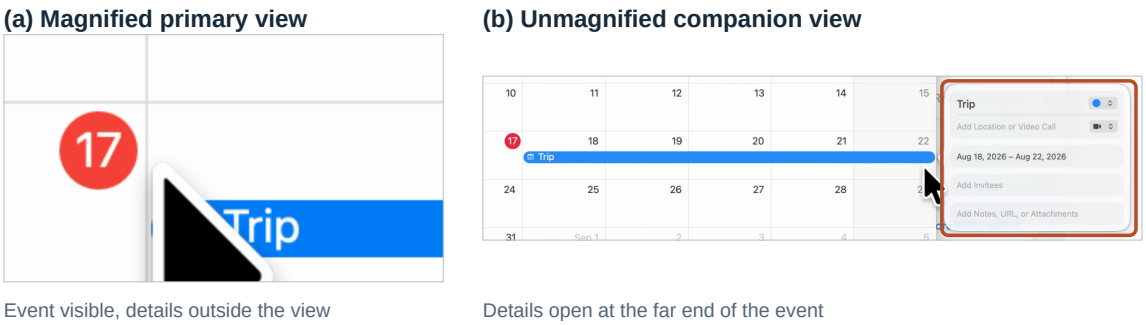



Fig. 2. Synchronized screen channels while P4 inspects an event in macOS Calendar. (a) The primary channel records the magnified view, showing the event but not its details popover. (b) A crop from the unmagnified companion channel reveals the popover at the event's far end (outlined), outside the view in (a).

Interaction pattern propositions can describe conditional strategies. For example, ``Primarily navigates web pages using VoiceOver Quick Nav single-key heading jumps … When intermediate sections lack headings, the person falls back to linear arrow navigation.'' This proposition describes a strategy for reaching content when headings are unavailable. These examples come from GAM's outputs for participants in our user study (Section 5.1).

*3.1.2 Accessibility barrier propositions.* Accessibility barrier propositions describe any frictions or difficulties one encounters on a specific user interface (UI). Each is labeled with an associated UI (a native application name or webpage URL), identifies the affected elements on the UI, and describes how the problem affects or could affect the person's interaction, including consequences such as a blocked action, repeated attempts, or a prolonged search. For example, when a participant with low vision was placing a food order, GAM reported: ``Validation error appeared at the modal header outside the magnified view port, making Save unresponsive after repeated clicks''. Although the interface displayed an error message, it fell outside the participant's magnified viewport when feedback was needed, making the Save button appear unresponsive. For a participant with a motor impairment adjusting a budget filter, GAM surfaced ``The budget range slider lacks numeric inputs, forcing repeated corrective drag loops … to set price bounds.'' This connects the slider's demand for precise dragging to the participant's repeated struggle to set the desired range.

Figure 6 illustrates webpage changes generated for the validation-message and budget-slider barriers. Figure 2 shows a native-application example: the companion recording reveals a Calendar event popover that is absent from the participant's magnified view.

*3.1.3 Proposition fields.* Each proposition contains three fields, with an additional interface field for barrier propositions:

- *Description* states the inferred interaction pattern or accessibility barrier in natural language. The examples above are proposition descriptions.
- *Reasoning* explains how the model interprets the observations to support the description, including connections between the person's actions and the interface's responses.

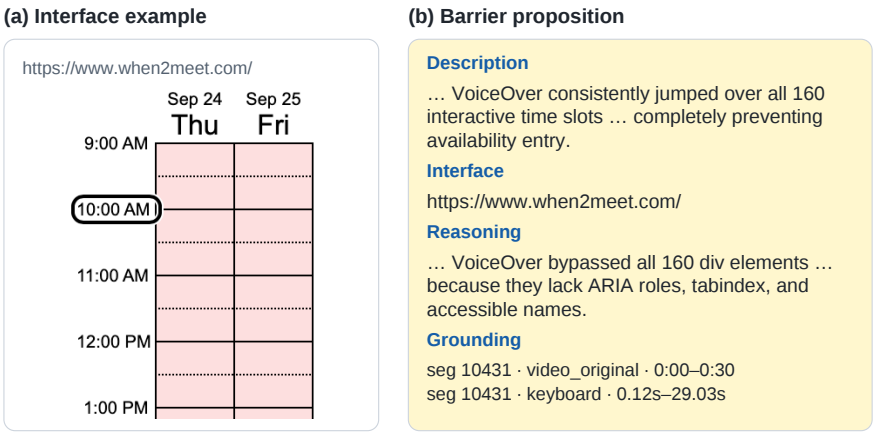



Fig. 3. A When2Meet barrier proposition. (a) A separate illustrative capture shows VoiceOver focus beside the availability grid. (b) Condensed fields from P7's session retain references to the original recording. Ellipses mark omissions.

- *Grounding* links the proposition to supporting observations. Each reference identifies a segment, an interaction source channel, and a timestamp or time range.
- *Interface (barrier propositions only)* identifies the webpage or native application in which the barrier occurs.

GAM maintains these propositions for retrieval and reuse. It groups barrier propositions by interface name, allowing downstream applications to retrieve the barriers relevant to a particular interface. Interaction pattern propositions remain available across interfaces.

Figure 3 pairs an illustrative interface view (a) with the fields of a barrier proposition (b), showing how the inferred difficulty connects to supporting evidence.

### 3.2 Constructing GAM

GAM observes and segments ongoing interaction, preprocesses and selects inputs, infers interaction patterns and barriers, and updates the understanding (Figure 4). Observation supplies ongoing evidence, while the processing loop operates on each segment. Each inference call combines the current segment's evidence with accumulated understanding and updates that understanding to interpret the next segment.

*3.2.1 Observing interaction traces.* GAM observes computer use as it unfolds, capturing multiple channels of interaction traces and recording their timing so they can be interpreted together. These channels provide complementary evidence of what the interface offers, how the person acts, and what feedback is available through their access methods:

- *Screen recordings.* GAM uses multi-channel screen recordings to capture two complementary views of the same interaction. The primary channel records the screen as presented, including macOS Zoom magnification and AT panels and overlays. This provides evidence for inferring what information the person could access through their current view. The companion channel supplies an additional screen-recording channel without any visual rendering changes by AT (e.g., magnifier) and with detected AT overlays excluded. In our current prototype, this second companion video recording channel is especially helpful for observing magnifier users,

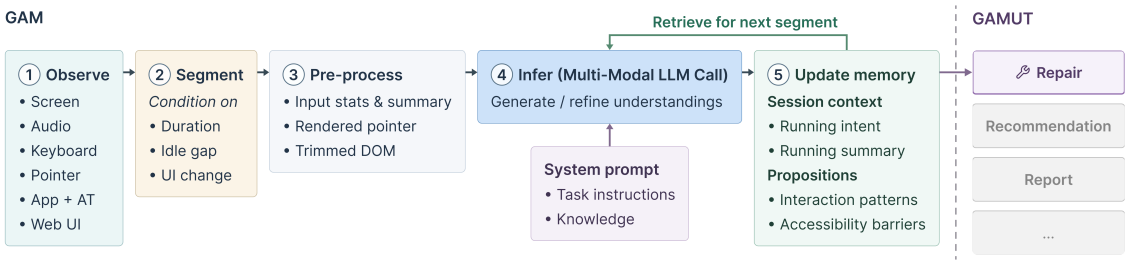



Fig. 4. GAM's processing loop, read from left to right: observe interaction, segment observations, preprocess inputs, infer, and update memory. The system prompt and retrieved understanding inform each inference call. Repair is shown connected to GAM's shared understanding. Greyed modules below illustrate alternative applications that can use the same representation.

as it captures the unmagnified, comprehensive screen view, and comparing it with the primary channel can reveal information outside a magnified viewport.

Figure 2 shows the two synchronized views, and Figure 4 shows their place in the processing loop. Their selection is described below.

- *System audio.* System audio captures feedback such as VoiceOver speech, allowing GAM to relate spoken output to interface content and user actions. Audio can also be compared with captions and visible feedback when the person relies on alternatives to sound. Available system audio is embedded with the primary channel of the screen recording.
- *Keyboard and pointer input.* Keystrokes, pointer movement, clicks, scrolling, and dragging actions are recorded, revealing attempted inputs and their timing, speed, and frequency. Capture at the operating system level preserves AT commands that may be consumed before reaching the foreground application. These traces help distinguish and verify user actions alongside the screen recording, or could be used to generalize recurring typing or pointer action patterns.
- *AT context.* GAM queries operating system state and uses heuristics to detect active AT, such as VoiceOver and macOS Zoom. This context guides the selection of interaction traces and acts as an important piece of information helping the multimodal LLM interpret user actions. These signals identify detected tools but do not provide a complete account of the person's access needs.
- *Interface context.* GAM records foreground application information and, during browser use, webpage URLs to identify the interface the person is currently using. This also determines segment boundaries.
- *Web UI, when available.* A browser extension captures DOM snapshots and a DOM-derived accessibility tree containing elements' roles and names. These provide structural, source-code level context for interpreting interface controls and grounding observations in specific elements. Figure 4 groups these structural signals as Web UI.

*3.2.2 Segmenting observations.* To process the ongoing stream of interaction data, GAM groups observations into bounded *segments*. A segment is the unit of new interaction evidence prepared for an inference call. This limits the new data sent to each inference call.

Segments close when they reach a configurable maximum duration, $T_{\max}$, or when the foreground application or webpage changes, including navigation within the same website. With keyboard and pointer capture enabled, segments

also close after $T_{\text{idle}}$ without input. The corresponding recordings, traces, and interface context form the segment's evidence (Figure 4).

Our prototype uses defaults of $T_{\text{max}} = 30$ seconds and $T_{\text{idle}} = 10$ seconds. These values aim to retain enough context to interpret related actions while providing regular opportunities to update the representation. Both thresholds are configurable. Future versions could adapt them to the person's interaction pace, the model's context capacity, or the application's latency requirements. Recording continues while completed segments are queued and processed in order. The time until an update becomes available therefore depends on both segment duration and delays from queueing and model inference.

*3.2.3 Preprocessing and selecting inputs.* For each segment, GAM first transforms or augments interaction traces into inputs that expose useful temporal, spatial, and structural information. It then selects the available evidence for the inference request (Figure 4).

*Channel preprocessing.* Preprocessing reduces repetitive input and makes interaction characteristics explicit without treating them as evidence of a barrier on their own:

- *Keyboard inputs.* Consecutive printable keys are grouped into typing bursts, while shortcuts and navigation commands remain timestamped entries. A summary is generated (through rule-based methods) to describe typing rate, commands, and shortcuts. These help the model relate input pace and command sequences to the interaction, without counting or reconstructing every event from the raw keystrokes input.
- *Pointer visualization and statistics.* GAM renders a separate video of pointer trails, click markers, and drag paths on the segment timeline. Deviations in the trail are visually amplified to make small movements easier to inspect. Numerical summaries computed from the original coordinates include movement speed and travel, pauses, button-hold durations, repeated clicks, and drag and re-grab counts. This provides a visual understanding of pointer actions, while the statistics supply temporal and quantitative context. After preprocessing, GAM replaces the raw pointer event listing with the pointer visualization video and statistical summary.
- *Web UI.* GAM trims available DOM input by removing scripts, styles, comments, and other nonsemantic content. The remaining markup retains page structure, text, and semantic attributes, subject to size limits, and is paired with the unchanged accessibility tree. This provides essential Web DOM while using fewer input tokens than the complete page source.

*Selecting inputs for each segment.* GAM uses deterministic rules to select input channels based on the available interaction traces. This step aims to supply the evidence needed to interpret each segment, avoiding unnecessary inputs to the inference call.

In our prototype, this approach is implemented through selection of the companion, unmagnified screen channel. GAM includes this channel only when the segment's recorded AT context indicates magnification, such as macOS Zoom. In this case, the broader view provides additional context for interpreting what falls within or outside the person's magnified viewport.

The prototype includes the other available interaction channels, subject to the preprocessing rules and input limits above. AT context alone does not establish whether these channels are needed: a person with a motor impairment may use no AT, while another person may combine several access methods. Web UI data remain conditional on browser capture. Future versions could extend the selection rules to use additional information about the task and interaction, or use model assistance to determine which channels are needed.

*3.2.4 Inferring interaction patterns and barriers.* For each segment, GAM makes a multimodal inference call that combines the current segment's interaction data with accumulated understanding. The request has three components (Figure 4).

*System prompt.* The system prompt combines two components that provide complementary guidance for interpreting each segment:

- *Task instructions.* These define GAM's role as an analysis engine over sequential segments of computer use, the two proposition types, and the description, reasoning, and grounding each proposition carries. They direct the model to relate what the person attempted to how the interface responded and what feedback reached them. They also specify how to revise propositions from earlier segments instead of restating them, and how to return the updated session context, including the person's current goal. Throughout, every material claim must cite an observation, and the reasoning must separate what was observed from what is inferred.
- *Pre-set accessibility knowledge.* A predefined knowledge library is maintained in an external JSON document and incorporated into the system prompt. Drawing on accessibility guidance, AT documentation, and research on access practices, it provides background on AT capabilities, interaction strategies, signals of difficulty, and potential support. The library covers screen reader use, low vision, hearing access, and motor interaction. The model is instructed to apply this knowledge only when supported by the observations, using it to interpret the person's interaction rather than assigning needs or solutions from a user category alone.

*Accumulated understanding.* Understanding what the person appears to be trying to achieve helps interpret their actions, the difficulties they encounter, and the outcomes they may expect. GAM therefore supplies four forms of accumulated context alongside each segment:

- *Current goal or intent.* A concise, revisable account of the person's apparent task provides context for interpreting their immediate actions. This is inferred from interaction, not treated as a confirmed statement of the person's goal. Figure 5a illustrates an update from locating an application to checking Calendar dates in the context of flight planning.
- *Running interaction summary.* A rolling summary of activity so far carries earlier context into the current segment. It compresses past observations to connect actions that span segment boundaries without resending the complete interaction history. Figure 5b shows a summary incorporating a transition from reading page content to encountering images without useful text alternatives.
- *Interaction pattern propositions.* Accumulated patterns describe ways the person operates interfaces, allowing strategies observed in earlier tasks to inform interpretation of the current interaction.
- *Accessibility barrier propositions.* GAM retrieves barriers associated with the current website or native application. This keeps prior difficulties relevant to the interface being interpreted and supports revising an existing barrier when further evidence appears.

For subsequent inference, GAM supplies compact proposition records centered on their descriptions, with identifiers and relevant metadata. Earlier reasoning and grounding remain stored but are not repeated in the request. This avoids repeating prior explanations and evidence references and keeps revisions focused on the current observations. The running summary and intent are refreshed after inference, carrying compressed context from earlier interaction into the next segment.

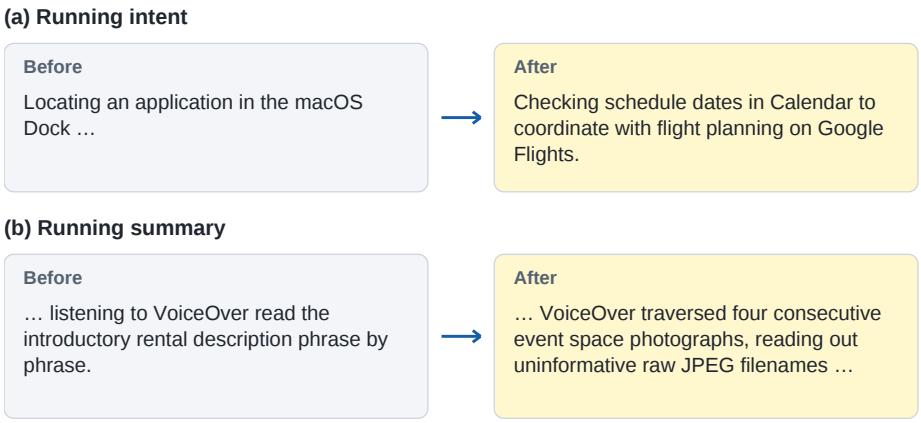



Fig. 5. GAM updates (a) running intent from locating an application to checking Calendar dates, and (b) running summary from reading rental information to encountering photographs with uninformative filenames. These separate examples are condensed from the GAM history interface. Ellipses mark omissions.

*Current segment evidence.* The selected recordings, formatted traces, numerical summaries, and AT and interface context are supplied with device information, audio availability, and the reason the segment ended. The model interprets these observations together with the accumulated understanding above. For example, a message outside a magnified view becomes relevant when considered alongside repeated activation of the associated button and the apparent task goal. The prompt requests channel and time references so that the resulting propositions can be checked against the observations.

*3.2.5 Updating the representation.* The final stage of each loop updates accumulated understanding from the new inference result (Figure 4). GAM adds new propositions and applies revisions to existing records using their identifiers. When revised descriptions or reasoning are supplied, they replace the previous values. New grounding references are appended, preserving earlier evidence. Associated metadata is updated when revised values are provided, and unchanged fields remain available. The running summary and current intent are refreshed with the returned context.

Figure 5 illustrates updates to intent and summary using two separate examples. The updated representation is then available to downstream applications and to the next inference call. Subsequent requests receive the compact interaction patterns, barriers associated with the current website or native application, and refreshed summary and intent. This closes the loop between observing new interaction and revising the account of the person's needs.

GAMUT applications use the updated descriptions, interface contexts, and supporting observations to generate repair plans, recommendations, and reports (Section 4). The repair application generates executable code through an additional model call.

### 3.3 Implementation

We implemented GAM as a macOS application communicating with a Python backend, complemented by a browser extension for Web UI capture. macOS provides native accessibility tools such as VoiceOver and Zoom across both native and web applications, allowing the prototype to observe different access methods in a shared desktop environment.

The native GAM application provides the session interface, and the backend coordinates system-level capture, segmentation, preprocessing, inference, and proposition storage. The extension supplies the web DOM and a DOM-derived accessibility tree.

The evaluated inference configuration uses Gemini 3.8 Flash with video sampled at eight frames per second, selected for its multimodal input support and inference speed. We use the configurable segmentation defaults described above. The prompt asks for a running summary of at most 300 words and a task intent of one or two sentences.

## 4 GAMUT: Applications of GAM

GAM's propositions give applications a shared account of the barriers a person encounters and how they interact with their computer. We demonstrate **GAMUT** (GAM UTilities), three prototype applications built on this representation.

The applications differ in scope and how directly they intervene. Accessibility repairs modify web interfaces to address observed difficulties. Feature recommendations suggest existing support that the person can choose to use across web and native applications. Personalized accessibility reports document access strategies and barriers across a session without changing the interface. We evaluate generated repairs in Section 5.4.

### 4.1 Accessibility Repairs

Prior systems have supported shared accessibility scripts [4] and LLM-generated webpage changes from natural language requests [23]. Our repair application uses GAM's propositions to generate targeted changes to webpages that resolve or mitigate barriers a person encounters. Barrier propositions identify the difficulty, the relevant webpage, and the affected elements, providing the application with a concrete target for the change. Interaction pattern propositions provide additional context about the person's recurring AT use and access strategies.

The repair application uses these propositions to formulate a repair plan. It presents a description of the proposed repair and lets the person request its generation. It combines the repair plan, barrier description, interaction pattern propositions, and captured webpage context in an LLM call that generates JavaScript to modify the affected elements. The person can then enable the repair through the browser extension, which applies the changes to the page only after it is enabled. The extension retains the repair and the person's choice, reapplying enabled repairs on later visits and allowing them to be disabled (Figure 6).

The application can generate several kinds of changes, two of which are illustrated in Figure 6. Prior work has adapted webpage presentation for low vision [1], restructured HTML for screen reader navigation [55], and generated interfaces fitted to motor abilities [13]. For the web validation barrier described in Section 3.1, a repair can position a validation message beside the button the person repeatedly activates. The Uber Eats example illustrates this placement, although its generated message contains an incorrect quantity. Bringing related feedback into the current view responds to the limited viewport available under magnification [53]. A scheduling page may prevent a blind person from selecting availability because its grid does not expose controls that can be operated with a screen reader. A repair can add labels and keyboard interaction to the time slots and communicate changes in selection state, following established guidance on keyboard operation and control semantics [52]. For a person with a motor impairment who struggles to set a price range by dragging a slider, a repair can add numeric fields for entering the minimum and maximum directly, with code to update the page's existing price filter. These examples link each proposed change to the difficulty recorded in a barrier proposition.

Beyond reuse on the same webpage, we envision applications that adapt repairs to other pages with matching barriers and elements, and communities in which people share repairs that others can reuse or adapt to their own access needs.

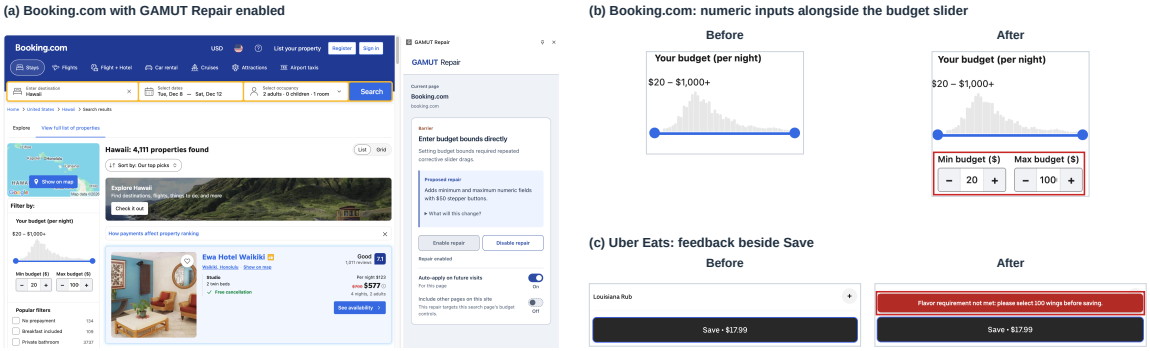



Fig. 6. Saved repairs shown in later captures. (a) Booking.com and the GAMUT panel with repair controls and future-visit settings. (b) Budget controls before and after adding numeric inputs and steppers. (c) Uber Eats before and after adding feedback above Save. Red outlines mark additions. The Uber Eats message incorrectly requests 100 wings for a 10-piece item, illustrating that improved placement does not ensure correct content.

### 4.2 Accessibility Feature Recommendations

Some barriers can be addressed or alleviated through support already available in operating systems, applications, or AT. A person may nevertheless be unaware of a feature or when it would help. Wu et al. [54] explored recommending accessibility features from observed mobile-device use, including reading distance, listening volume, and near-miss button presses. GAM's propositions give a recommendation application context for connecting this support to an encountered difficulty and the person's established access methods.

The recommendation application uses GAM's interaction pattern and accessibility barrier propositions to suggest relevant existing features and interaction techniques. In our prototype, it presents these recommendations in the native application, explaining why each may help and how to enable or use it. For example, when a VoiceOver user encounters photographs without useful text alternatives, a recommendation can explain how to invoke VoiceOver's image description feature. For a person using magnification who has difficulty locating dates in a calendar, it can suggest entering departure and return dates directly into the existing fields (Figure 7). The recommendation connects the barrier to a way of using existing support, which the person can also draw on in similar situations during later tasks.

### 4.3 Personalized Accessibility Reports

Repairs and recommendations provide support during use. A reporting application can use the same representation to document the person's experience. Prior work such as WebinSitu used recorded browsing behavior to examine accessibility difficulties [3]. GAM's propositions provide material for a personalized accessibility report: interaction patterns describe the person's access methods and strategies, while barrier propositions document the difficulties, affected interfaces, and supporting observations.

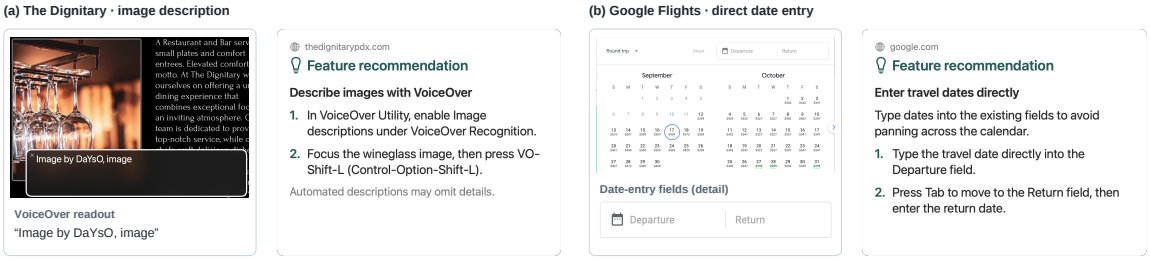



Fig. 7. Two illustrative recommendation examples, (a) on the left and (b) on the right, pair interface context with a native recommendation card. (a) An uninformative photo description on The Dignitary motivates guidance for invoking VoiceOver image descriptions. (b) Google Flights date selection motivates guidance for typing into the existing Departure and Return fields. Each card identifies the website and explains the feature and how to use it.

Our reporting application uses GAM’s propositions and their grounding references to generate a report of the session through an LLM call, associating each barrier with its webpage or application. An application could use the recorded URLs to assemble a report for a particular website. The report summarizes the person’s interaction patterns, then lists accessibility barriers with affected elements, supporting evidence, relevant WCAG references, and suggested implementation changes (Figure 8). The report presents interface problems in the context of a particular person’s use, connecting the inferred difficulties to their supporting observations.

The HTML example links findings to playable clips retrieved from GAM’s stored recordings through the propositions’ grounding references. Figure 8 shows one finding from an edited report generated offline from the study recordings.

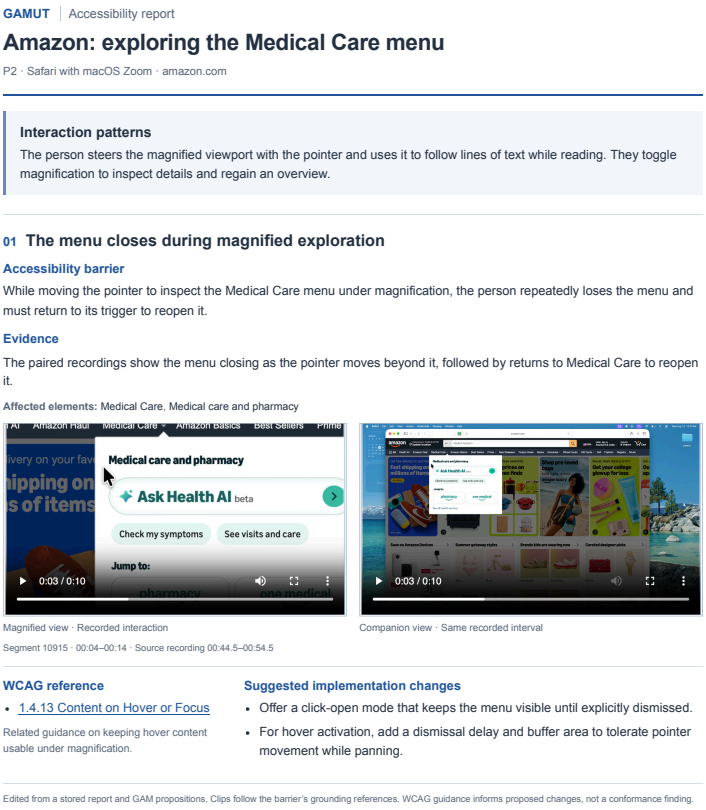



Fig. 8. An edited HTML report for P2's Amazon task. The report connects pointer-guided magnification with repeated dismissal of the Medical Care menu. It includes affected elements, paired magnified and companion clips retrieved through the barrier proposition's grounding references, related WCAG guidance, and suggested implementation changes. This presentation was prepared after the study from a stored report and propositions.

## 5 Evaluating GAM

We evaluate GAM based on *(i)* its performance in surfacing accessibility barriers from recordings of usage and *(ii)* the effectiveness of interventions aimed at addressing these barriers. We recruited participants to perform various computer tasks selected to present potential accessibility barriers relevant to their access needs. We measure recall and proposition accuracy using outputs generated from the recorded datasets and test the resulting repairs on live websites. Participant ratings and qualitative feedback concern outputs presented during the study.

The full-observation and static-input ablation conditions are described in Section 5.3.

### 5.1 User Study Methodology

*Participants and preparation.* We recruited ten participants from nearby cities through an institutional participant pool: four blind participants, four with low vision, and two with motor impairments. Some reported overlapping access needs (Table 1). Sessions were scheduled for three hours, with compensation of $25 per hour. Before each session, we

Table 1. Participants' reported access needs, familiar assistive technologies, and study setup.

| Participant | Reported access needs | Familiar assistive technologies | Study setup |
|---|---|---|---|
| P1 | Blind | NVDA on Windows | VoiceOver |
| P2 | Low vision | Windows Magnifier and display scaling | macOS Zoom |
| P3 | Blind | Primarily JAWS, occasional NVDA | VoiceOver, own keyboard |
| P4 | Low vision | macOS Zoom, Smart Invert, Speak Selection | macOS Zoom, Smart Invert, Speak Selection |
| P5 | Low vision | Zoom on iPhone and iPad | macOS Zoom |
| P6 | Low vision, motor, and cognitive | Zoom on iPhone | macOS Zoom |
| P7 | Blind | NVDA on Windows | VoiceOver |
| P8 | Motor | Sticky Keys, Speak Selection | Sticky Keys, Speak Selection, own mouse |
| P9 | Motor and cognitive | None | Standard macOS setup |
| P10 | Blind | JAWS on Windows | VoiceOver |

asked about the participant's access needs, usual hardware, software, and assistive technologies. Participants could bring preferred peripherals.

*Task selection.* We developed a task pool around barrier categories identified in accessibility studies, surveys, and community discussions [13, 31, 34, 35, 49]. We audited real websites and native applications to identify scenarios exposing these barriers, documenting each task's goal, access method, and affected interface elements. From this pool, we selected six to eight tasks based on each participant's reported access needs and usual technologies, adapting them as needed.

*Procedure.* Sessions used a macOS laptop configured with the participant's preferred AT settings. When their usual software was unavailable, we used an available alternative, such as macOS Zoom for Windows Magnifier. For participants unfamiliar with VoiceOver, we provided the official VoiceOver tutorial on macOS and warm-up sessions to familiarize them with the controls needed for the tasks.

For each task, the researcher introduced the goal and started recording. Participants used real websites and applications and could follow their own strategies, encounter unanticipated difficulties, or work around the audited barrier. Afterward, we discussed the difficulties they experienced, their severity, and the support they wanted. The researcher read GAM's propositions aloud, answered questions, and collected participants' feedback through ratings and elaborations. When available, participants also tried, discussed, or rated repairs and feature recommendations from the GAMUT applications.

A final interview covered how well the tasks reflected their usual difficulties and what support they would want. We recorded interactions and discussions with consent. Some participants did not complete all assigned tasks. Appendix A lists the study activities.

### 5.2 Datasets

*User-study recordings: GAM-Eval-US.*. The study yielded 58 task recordings from ten participants. Each recording covers one participant performing one task and synchronizes the available screen video, audio, keyboard and cursor traces, application and AT context, and web DOM and accessibility tree snapshots. From the task recordings and

discussions, we documented 104 barriers across 54 of the 58 recordings. One researcher finalized severity labels from the participants' judgments and discussion: 42 major and 62 minor barriers.

*Reproduced barriers: GAM-Eval-RB..* Eight of our ten participants were blind or had low vision. To extend the evaluation's coverage beyond this participant sample and scale up the size, we complemented the user-study recordings with researcher recordings from the audited scenario pool. These cover blind, low vision, motor, and deaf or hard of hearing (DHH) access needs, including DHH scenarios absent from the user study. Researchers performed these tasks using the relevant AT, recording one to three interaction pathways per scenario. The resulting dataset contains 62 recordings across 34 scenarios and 18 barrier categories, with 64 documented barrier instances. Appendix B.1 summarizes their coverage.

### 5.3 Evaluation Metrics for Propositions

We assessed barrier recall, proposition accuracy, and participant feedback on propositions presented during the study. One researcher reviewed recall matches and assessed proposition accuracy.

*Full observation and static input.* We evaluated GAM on both recorded datasets under full observation and static input. Full observation used the available interaction channels in GAM's standard pipeline. Static input retained application and AT context and available web structure, while omitting video, audio, and input traces.

*Barrier recall.* A documented barrier was recalled when at least one proposition described the same underlying problem and affected interaction. The reviewer manually checked matches under both configurations. Each documented barrier counted once, regardless of the number of matching propositions. We report recall over all 64 reproduced barrier instances and all 104 user-study barriers, with a separate analysis of the 42 major barriers. Percentages are pooled over barriers within each dataset or access group. We compare user-study participants' full and static recall with a two-sided exact sign test, using the participant as the unit. Interaction patterns have no defined exhaustive reference set and were not assessed for recall.

*Proposition accuracy.* A proposition can describe a pattern or barrier that was not documented in the reference set, so lack of a match does not establish an error. We therefore reviewed all 106 unmatched full-observation barrier propositions and all 116 interaction pattern propositions in the blind, low vision, and motor cases across both datasets. The reviewer assessed whether each account was supported by its recording and session context, recording a binary judgment and explanatory notes. The 129 matched barrier propositions were outside this review cohort. A recall match establishes correspondence with a documented barrier without verifying every detail of the proposition.

*Participant feedback on propositions.* Participants rated how much they agreed that each proposition generated by GAM accurately described their experience in the study, from 1 (strongly disagree) to 7 (strongly agree), and explained their ratings. These ratings concern outputs presented during the study, while recall and accuracy concern outputs generated from the recorded datasets.

*Exploratory participant profiles.* We also evaluated GAM on P8's and P9's recordings with short researcher-written profiles of their conditions, input methods, and access difficulties, assessing recall against the same 20 documented motor barriers. The profiles supplied additional background information, which was used to guide proposition generation alongside interaction evidence. They drew on pre-study information exchanged with the participants.

### 5.4 Evaluation Metrics for Interventions

We focused on web repairs generated by GAMUT from GAM's representation, evaluating whether they resolved the associated barriers and helped improve participants' experiences.

*Technical barrier resolution.* A researcher tested all 132 full-observation repairs from blind, low vision, and motor cases: 74 from GAM-Eval-US and 58 from GAM-Eval-RB. The researcher applied each unchanged patch on the live website, reproduced the recorded barrier, and tried the affected interaction using relevant AT. A repair counted as *resolved* when the targeted function became accessible, and *not resolved* when the core difficulty remained. If website changes prevented reproducing the barrier, the repair was *not testable*.

*Participant feedback on repairs.* Participants rated how much they agreed that repairs presented during the study helped make their experience more accessible or made the task easier to complete, from 1 (strongly disagree) to 7 (strongly agree). They explained what remained unresolved and described an ideal repair when the rating was below the maximum. We analyzed 30 ratings alongside the discussions. These included live-generated and previously generated repairs, with guided trials and assisted demonstrations. Recommendation feedback was analyzed qualitatively rather than pooled with repair ratings.

## 6 Results

We first report barrier recall, proposition accuracy, and participant feedback on propositions generated during the study. We then turn to the interventions: whether generated repairs resolve their barriers, and how participants judged the repairs and recommendations they tried.

### 6.1 Understanding Accessibility Needs

*GAM recovered most GAM-Eval-RB barriers and a smaller share of GAM-Eval-US barriers.* Under full observation, GAM recovered 61 of 64 GAM-Eval-RB documented barrier instances (95%, Figure 9), with 100% recall for the blind, low vision, and DHH recordings and 9 of 12 for motor recordings. GAM-Eval-RB pathways were built around known barriers, so this result shows that GAM recognizes such barriers when a recording presents a focused encounter with them.

In GAM-Eval-US, GAM recovered 63 of 104 documented barriers (61%). Participants used workarounds and switched between ways of navigating, and some difficulties surfaced only in the discussion after a task, while their visible evidence was brief or scattered across the recording. These differences plausibly explain part of the lower recall, although the datasets also differ in tasks and barrier composition, so the comparison does not isolate one cause.

Motor barriers had the lowest recall in both datasets: 9 of 12 GAM-Eval-RB instances and 6 of 20 GAM-Eval-US barriers. Unlike a keyboard trap or repeated panning, motor effort can be hard to infer from an action that eventually succeeds but remains difficult for the person to perform. GAM recovered P8's difficulty with a budget slider, but missed the reported difficulty typing longer passages and using the Speech menu after an unfamiliar Speak Selection shortcut failed. Observing an action does not by itself reveal its cost to the person. Some difficulties also concerned the meaning of a control. P5 hesitated over a close control because she could not tell whether it would remove one item or discard the whole cart, and GAM did not report that uncertainty.

*Recall was higher for barriers labeled major.* GAM recovered 33 of the 42 GAM-Eval-US barriers labeled major (79%) and 30 of the 62 labeled minor (48%). Recovered major barriers were typically sustained obstacles to a central action,

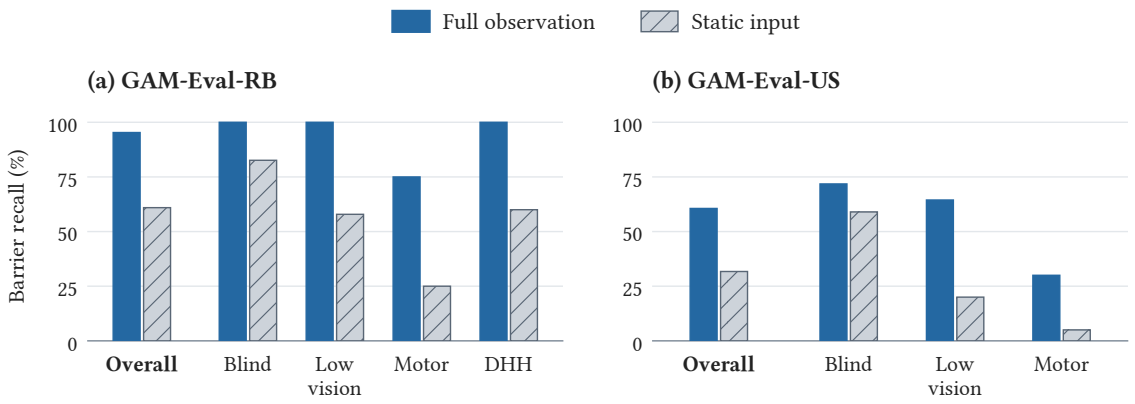



Fig. 9. Recalled/documented barriers by dataset and access group. Bars show pooled percentages for full observation and the static input ablation. Exact counts appear in Appendix B.

for example P7's inability to select availability slots in When2Meet with VoiceOver and P5's repeated attempts to save a food order while a required-selection message sat outside the magnified viewport. Missed minor barriers were typically brief detours, for example the extra effort of locating a Save control and repeated clicks on a quantity stepper. A difficulty that persists during a task may leave clearer evidence for GAM and be more likely to be labeled major. Higher recall for major barriers therefore does not, by itself, show which interaction signals helped GAM detect them.

*Most reviewed propositions were judged accurate.* Across the blind, low vision, and motor cases of both datasets, GAM produced 235 barrier propositions under full observation, of which 129 matched a documented barrier. In the review of the remaining 106 barrier propositions and all 116 interaction pattern propositions, 215 of 222 (96.8%) were judged accurate. These comprised 106 of 106 unmatched barrier propositions and 109 of 116 interaction pattern propositions (Table 9).

*Participant feedback on propositions was generally positive.* Participants rated how much they agreed that GAM's propositions accurately described their experience, from 1 (strongly disagree) to 7 (strongly agree). Average ratings were 6.65 for barrier propositions and 6.27 for interaction pattern propositions (Figure 10).

During a food ordering task, GAM identified why P5 kept pressing Save without seeing a response: a message indicating an incomplete flavor selection appeared outside her magnified viewport. P5 confirmed this explanation: "when I pressed Save, I kept pressing, and I'm like, 'Nothing changed.'" She rated the description 7 of 7. This illustrates the distinction introduced in Section 3: a message can exist in the interface without being available to the person at the moment it matters.

*Lower ratings reflected unrecognized events and misread intentions.* P2 gave a barrier proposition about size-chart panning a rating of 4: it noticed the movement but missed that he had switched tabs and was deliberately checking new measurements. He rated a description of an unresponsive store-finder click 1 because he did not recall the event, and clarification brought acknowledgment but no revised rating. Even a rating of 5 for a menu description came with a correction: the repeated movement was a demonstration, not failed navigation. These explanations qualify

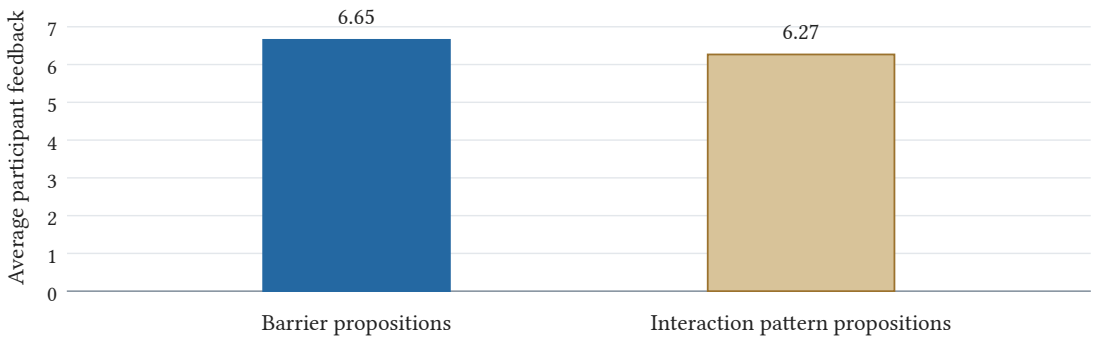



Fig. 10. Average participant feedback on barrier propositions (49 ratings) and interaction pattern propositions (15 ratings) generated during the study. Ratings indicate agreement that the propositions accurately described the participant's experience, from 1 (strongly disagree) to 7 (strongly agree).

what positive participant feedback establishes. Participants could recognize a broad pattern while correcting its cause, purpose, or significance.

*Propositions vary in significance.* GAM produced ten barrier propositions while P7 booked a flight. He confirmed them but explained that "all those issues I was able to get past for what I needed for this task, except for the calendar." The rest "were irritating, they were annoying, but they weren't a barrier." He also qualified that the child-ticket stepper would have blocked a task that required booking for children. Several propositions also described related issues: two concerned missing page structure, two concerned focus resetting after an update, and two concerned the same terms-of-sale control. These observations distinguish positive feedback on an account from the importance and organization of the issues it describes.

*Full observation recovered more barriers than static input.* Static input recovered 39 of 64 GAM-Eval-RB instances (61%) and 33 of 104 GAM-Eval-US barriers (32%), against 61 of 64 (95%) and 63 of 104 (61%) under full observation. In GAM-Eval-US, full observation had higher recall for eight participants and equal recall for two (two-sided exact sign test, $p = .008$). Table 8 gives the per-participant counts.

The gap was largest for low vision: 29 of 45 GAM-Eval-US barriers with full observation against 9 of 45 with static input. Interface structure can establish the static evidence that a control and a message exist, but only the recording shows dynamic interactions indicating whether the message entered the magnified view and how the person responded. For blind participants the gap was smaller (28 of 39 against 23 of 39), consistent with more barrier evidence could be collected from interface code alone. Motor recall was 6 of 20 against 1 of 20, indicating reliance on interaction data.

*Participant profiles increased motor-barrier recall in an exploratory comparison.* Supplying participant profiles increased recall for P8 and P9 from 6 of 20 documented barriers (30%) to 10 of 20 (50%). P8's recall increased from 1/7 to 4/7, and P9's from 5/13 to 6/13 (Table 2). The seeded runs recovered five previously missed barriers but missed one that the unseeded run had recovered. New matches included P8's difficulty typing longer passages and using the Speech menu, and P9's difficulty locating a hidden sign-in form and resizing an image.

Table 2. Exploratory motor-profile comparison using the same recordings and documented barriers. Each cell reports recalled barriers over documented barriers.

| Participant | Full observation | Full observation with profile |
|---|---|---|
| P8 | 1/7 | 4/7 |
| P9 | 5/13 | 6/13 |
| Total | 6/20 (30%) | 10/20 (50%) |

Table 3. Technical resolution of the targeted barrier for all generated repairs, tested on the live site with the relevant AT. Percentages use all generated repairs in each row. Not-testable repairs are reported separately from failures.

| Dataset | Group | Generated | Resolved *n* (%) | Not resolved | Not testable |
|---|---|---|---|---|---|
| GAM-Eval-RB | Blind | 34 | 28 (82.4%) | 4 | 2 |
| GAM-Eval-RB | Low vision | 15 | 11 (73.3%) | 4 | 0 |
| GAM-Eval-RB | Motor | 9 | 5 (55.6%) | 4 | 0 |
| GAM-Eval-RB | All | 58 | 44 (75.9%) | 12 | 2 |
| GAM-Eval-US | Blind | 50 | 40 (80.0%) | 10 | 0 |
| GAM-Eval-US | Low vision | 19 | 15 (78.9%) | 4 | 0 |
| GAM-Eval-US | Motor | 5 | 4 (80.0%) | 1 | 0 |
| GAM-Eval-US | All | 74 | 59 (79.7%) | 15 | 0 |

*Model processing can delay updates.* The 227 full-observation GAM-Eval-RB model calls had a median duration of 46.4 s (interquartile range 30.9–65.3 s), and the 644 GAM-Eval-US calls had a median of 50.5 s (34.0–75.2 s). The 95th percentiles were 123.8 s and 132.0 s, respectively. These durations exceed the nominal 30-second segment cap and show that model processing can delay updates while observation continues.

### 6.2 Generating Interventions

*Most generated repairs addressed their tested target.* The repair application generated 58 GAM-Eval-RB and 74 GAM-Eval-US repairs under full observation (Table 3). Researcher testing judged 44 of 58 GAM-Eval-RB repairs (75.9%) and 59 of 74 GAM-Eval-US repairs (79.7%) as resolving their targeted barrier. Overall, 103 of 132 generated repairs (78.0%) received a positive judgment. Twenty-seven were judged not resolved, and two could not be tested because the website had changed. Among the 130 testable repairs, 103 (79.2%) were judged resolved.

Successful repairs added headings for screen reader navigation, provided numeric inputs as alternatives to sliders, and placed quantity controls beside their labels within the magnified view. Unsuccessful tests often reported no relevant change or controls that still could not perform their intended action. For example, repairs intended to add headings or landmarks left those navigation options unavailable to VoiceOver. Two carousel repairs added pause buttons that the reviewer found ineffective and difficult to locate. Another repair displayed incorrect size information in an add-to-cart confirmation. These failures motivate checking that repairs change the intended interface elements, perform the intended action, and provide accurate feedback (Section 7.3).

*Participants found repairs useful, but improvements were often incomplete.* Across 30 repair-helpfulness ratings from nine participants, the mean was 5.8 and the median 6 on a scale from 1 (strongly disagree) to 7 (strongly agree), and 26 ratings were above the midpoint of 4. These ratings expressed agreement that the repair helped make the participant's experience more accessible or made the task easier to complete. Thirteen ratings concerned live repairs,

14 previously generated repairs, and three repairs of uncertain provenance, and both independent and assisted trials are included. These ratings concern repairs presented during the study. Technical testing concerns repairs generated from the recorded datasets.

P7's live Craigslist repair shows how a structural change helped with the task: "having the headings allows me to actually get to the thing I'm trying to find." He rated it 6. He described the live When2Meet repair as making an inaccessible availability grid operable: "It goes from unusable to usable, so it is definitely helpful," and rated it 5. Yet he wanted to traverse time slots within a day rather than move across days at one time slot, and the separate event-creation calendar still lacked confirmation of the selected dates. Access improved while the interaction still differed from what he needed. He rated the completed live Fly Southern repairs 4: the fare headings and the terms checkbox improved, but he still could not change the flight date or finish the task on his own, and some repairs were still generating during the trial.

The scope of a repair could also be narrower than the repeated barrier on the page. In P1's scheduling task, a repair made his availability grid operable while the adjacent group grid remained unchanged.

Other participants described improvements that left feedback or recovery unaddressed. P1 rated a live product-stock repair 6 but explained that it still did not announce which size he had selected. P10 rated a previously generated When2Meet repair 3 after an assisted trial: he could reach named availability cells but still wanted selection instructions and confirmation that a selection had registered. For P6, a live repair that made the cart's removal control more visible helped with activation but raised a further requirement once the cart reflowed: "what if I got rid of the wrong one? Can I get it back real quick". The desired outcome must include confirmation and recovery, not only an operable control.

A repair could also introduce new difficulty. P2 rated a live spacing repair 2 because moving controls "just kind of clutters things up a little more," while a previously generated contrast repair for the same page received "6 or 7," coded as 6.5. A change intended to address one barrier can add a new visual cost. P2 described function and design as working "hand-in-hand". He wanted familiar visual conventions, such as a red X or trash icon, to retain the meaning of a removal control. P6 similarly asked for a control that was "bold or red."

*Recommendations helped when they fit the task and existing workflow.* The selected recommendation trials showed both useful support and limits. P3 preferred the richer detail of a generated image-description repair to a VoiceOver description that was too general. P10 valued the keywords from VoiceOver Image Descriptions, while still wanting more information about the room. P4 welcomed entering a travel date by typing instead of panning across a magnified calendar. In contrast, P9 found that recommended arrow-key input did not operate a slider as described. P7 said "I think both are important" when comparing a recommendation and repair for the same task, rating both six. These selected cases do not establish a general preference or comparative effectiveness between repairs and recommendations.

*Participants wanted ways to guide and revise support.* Participants' requests extended beyond accepting a generated repair. P7 wanted to decide which issues to address: "It'd be cool if you could kind of say yes fix this, no don't fix this. Even just a yes or no," with "a little text input to give more information about an issue or even information generally about an issue that it didn't catch." He also valued suggestions because "sometimes you don't know what the issue is, and it maybe has a suggestion that you didn't think of." These requests point to input and correction alongside observation, without requiring the person to diagnose or specify every interface change.

*Privacy conditions shaped willingness to use ongoing observation.* P4 wanted assurance that bank details, card numbers, and Social Security numbers would not be collected or shared. He was comfortable with analysis that remained on

his devices. P2 asked where data sent to a model service would go and how long it would be kept, suggesting redaction or local models. P9 was not troubled provided that nothing identifying him was released. These individual conditions qualify interest in support that observes computer use continuously.

## 7 Discussion and Future Work

GAM turns observations of computer use into a shared account of a person's interaction patterns and the barriers they encounter, and repairs, recommendations, and reports draw on that account. We discuss how this representation and the applications built on it could develop through better organization, coordinated support, feedback, and broader use.

### 7.1 Building Organized Accounts of Propositions

GAM maintains individual propositions in flat lists, with barrier propositions grouped by interface. As propositions accumulate, downstream applications may find it increasingly difficult to identify relevant needs, particularly when many findings overlap or concern the same interface. A structured index could make relationships among propositions explicit and support selective retrieval.

A hierarchy could organize propositions from general access needs to activities such as shopping or productivity, then to specific tasks and interface encounters. For example, a general need to reduce precise dragging could connect difficulties with shopping filters and controls in productivity tools. More specific propositions would describe how that need arises in each context and retain links to supporting observations. Related barriers could also be linked by a shared element, cause, or remedy, allowing a repair application to address several instances together or a report to present them as one issue.

Metadata could then guide which propositions an application retrieves from this structure. Recency would indicate when a proposition was last observed or confirmed, while temporal scope would distinguish persistent needs from temporary ones. Expiration and urgency could indicate whether a need remains relevant and when support is required. Severity would describe whether a barrier blocks the current goal, forces a detour, or adds effort. Applications could use these attributes to select relevant propositions from the current task's branch, together with applicable general needs. This would provide a focused account for each application while preserving access to the broader context and supporting evidence.

### 7.2 Combining Interventions

GAMUT presents repairs, feature recommendations, and personalized accessibility reports as three applications of GAM's representation.

A repair directly adapts an interface to the person, but may be unreliable, difficult to apply across different website structures, or insufficient for some of the person's needs. An existing accessibility feature may offer support across applications, but adopting it can require learning, configuration, or a change of strategy, and its usefulness depends on the task. A report can preserve unresolved difficulties and their evidence when a local change or existing feature does not provide sufficient support. These applications could therefore play complementary roles as the person's context and needs change.

Which form of support to offer therefore depends on the person's interaction patterns and the situation [40], and the choice can be treated as an optimization problem. A system could estimate the expected benefit of each option for the person's goal against its costs, including adoption effort, disruption of a familiar workflow, the risk that a change breaks, and the uncertainty of GAM's own account, and offer the option or combination with the highest expected value. This

follows the mixed-initiative principle of weighing the costs and benefits of acting under uncertainty about the person's goals [16] and SUPPLE's formulation of interface adaptation as optimization against estimated effort [12, 13]. GAM's propositions could provide essential information for this choice: the barrier, the strategies the person relies on, and the interface conditions a repair would depend on. Other forms of support, such as teaching an interaction technique or handing a step to an agent, could draw on the same account and be selected in the same way.

### 7.3 Refining Understanding and Support with Feedback

Interaction provides evidence of accessibility needs, but an observed action does not fully establish the effort it required or what the person understood (Section 6.1). The exploratory seeded comparison (Section 6.1) and participants' requests for input and control (Section 6.2) motivate ways for people to guide and correct this account.

We envision an interface through which the person can complete or correct GAM's understanding at three levels. The first is background. A description of access needs could guide interpretation of the observed interaction. The second is the understanding itself. The person could flag a barrier GAM missed, correct one it described, or name an interaction technique it should account for, leaving a downstream application to propose support. The third is the support. The person could specify the change they want, as end-user programming and natural language customization let people build their own access programs and interface changes [15, 23]. The division of labor should follow what each side can provide. The person can explain what remains difficult and what they want, while GAM supplies evidence about the interface, actions, and AT context. Self-reported needs and observed evidence should stay distinguishable in the account, so that background guides interpretation without being treated as evidence that an event occurred.

Feedback could also close the loop after support is applied. GAM could observe whether the intended action succeeds, whether the person adopts the change, and whether it introduces new difficulty [25]. For repairs, verification should test the relevant page state and outcome, including whether feedback reflects a completed action and whether related controls remain usable. These checks could guide further revision when a repair falls short, alongside the person's assessment of remaining difficulty (Section 6.2). The resulting evidence could update GAM's propositions and inform the downstream application's next change. The person should be able to reject, adjust, or reverse a change without having to supervise every step.

### 7.4 Sharing Interventions Across the Community

An intervention that helped one person may help another who uses similar AT, shows similar interaction patterns, and encounters similar barriers on a related interface. GAM's account could help identify this correspondence. Each intervention could be stored with the barrier it addressed, the interaction patterns of the person it helped, the interface conditions it depended on, and how it turned out. Collaborative accessibility scripting and metadata authoring already let people share fixes for a page [4, 44]. Indexing a shared intervention by interaction evidence rather than by page alone could let a system recognize that a newcomer to a site shows patterns and barriers similar to someone the intervention helped, and offer it before much evidence about the newcomer has accumulated. This could ease the cold start that recommenders face [54]. A reused intervention would still need checking against the current interface and the person's own goal, and the person could adjust it as their use evolves. Aggregated across people, these records could also help developers identify recurring barriers across access methods. Sharing of this kind raises the disclosure questions discussed below.

### 7.5 Extending the Modalities

GAM currently observes screen recordings with system audio, keyboard and cursor traces, application and system metadata, and available web DOM and accessibility trees. Its inference relates what the interface exposed to what the person could perceive or operate, what they did, and what feedback they received. The same approach could extend to other access needs by adding channels that carry their interaction. For speech input, the recognized commands, misrecognitions, and corrections could show where an interface resists voice control in ways a screen recording alone does not explain. On mobile devices, GAM could use touch and gesture traces, such as those captured by runtime ability toolkits [21], to identify difficulties such as repeated taps on a control or unsuccessful attempts to drag an item. These additional channels could extend GAM's approach to more input methods and devices while retaining its approach of inferring accessibility needs from observed interaction.

### 7.6 Privacy

Participants' privacy concerns (Section 6.2) highlight the need for control over what GAM collects, shares, and retains. Reflecting on his study experience, P4 felt the system's benefits could outweigh some privacy risks, while still emphasizing protection of sensitive information and analysis on his own devices. The current prototype sends observations to a cloud model service and stores evidence locally, without a dedicated filter for sensitive content. Potential safeguards include excluding sensitive applications and websites, and locally redacting sensitive details or withholding affected segments before a model call. Local inference offers another promising direction, with evolving open-weight foundation models.

## 8 Conclusion

We presented the General Accessibility Model (GAM), which turns multimodal observations of computer use into revisable propositions about interaction patterns and accessibility barriers. GAMUT illustrates how downstream applications can use this shared representation for webpage repairs, feature recommendations, and personalized accessibility reports. Evaluation with reproduced barriers and recordings from ten participants showed that GAM can surface consequential difficulties and inform repairs that make targeted functions accessible. Participants' feedback further emphasized the need to preserve familiar strategies, provide confirmation and recovery, and support the person's broader task. Together, these contributions advance an approach to accessibility support grounded in how a person perceives, operates, and experiences an interface. Extending this approach over longer use will require keeping the understanding and resulting support aligned with changing needs, interfaces, and user feedback.

## References

[1] Jeffrey P. Bigham. 2014. Making the Web Easier to See with Opportunistic Accessibility Improvement. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, 117–122. doi:10.1145/2642918.2647357

[2] Jeffrey P. Bigham, Jeremy T. Brudvik, and Bernie Zhang. 2010. Accessibility by Demonstration: Enabling End Users to Guide Developers to Web Accessibility Solutions. In *Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '10)*. ACM, 35–42. doi:10.1145/1878803.1878812

[3] Jeffrey P. Bigham, Anna C. Cavender, Jeremy T. Brudvik, Jacob O. Wobbrock, and Richard E. Ladner. 2007. WebinSitu: A Comparative Analysis of Blind and Sighted Browsing Behavior. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '07)*. ACM, 51–58. doi:10.1145/1296843.1296854

[4] Jeffrey P. Bigham and Richard E. Ladner. 2007. Accessmonkey: A Collaborative Scripting Framework for Web Users and Developers. In *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility (W4A '07)*. ACM, 25–34. doi:10.1145/1243441.1243452

[5] Jeffrey P. Bigham, Irene Lin, and Saiph Savage. 2017. The Effects of "Not Knowing What You Don't Know" on Web Accessibility for Blind Web Users. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '17)*. ACM, 101–109. doi:10.1145/3132525.3132533

[6] Giorgio Brajnik, Yeliz Yesilada, and Simon Harper. 2010. Testability and Validity of WCAG 2.0: The Expertise Effect. In *Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '10)*. ACM, 43–50. doi:10.1145/1878803.1878813

[7] Giorgio Brajnik, Yeliz Yesilada, and Simon Harper. 2012. Is Accessibility Conformance an Elusive Property? A Study of Validity and Reliability of WCAG 2.0. *ACM Transactions on Accessible Computing* 4, 2 (2012), 1–28. doi:10.1145/2141943.2141946

[8] Deque Systems. [n. d.]. axe-core: Accessibility engine for automated Web UI testing. https://github.com/dequelabs/axe-core. Accessed July 2026.

[9] Morgan Dixon and James Fogarty. 2010. Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, 1525–1534. doi:10.1145/1753326.1753554

[10] Nadeen Fathallah, Daniel Hernández, and Steffen Staab. 2025. AccessGuru: Leveraging LLMs to Detect and Correct Web Accessibility Violations in HTML Code. In *Proceedings of the 27th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '25)*. ACM, 1–22. doi:10.1145/3663547.3746360

[11] Raymond Fok, Mingyuan Zhong, Anne Spencer Ross, James Fogarty, and Jacob O. Wobbrock. 2022. A Large-Scale Longitudinal Analysis of Missing Label Accessibility Failures in Android Apps. In *CHI Conference on Human Factors in Computing Systems (CHI '22)*. ACM, Article 461, 16 pages. doi:10.1145/3491102.3502143

[12] Krzysztof Gajos and Daniel S. Weld. 2004. SUPPLE: Automatically Generating User Interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI '04)*. Association for Computing Machinery, New York, NY, USA, 93–100. doi:10.1145/964442.964461

[13] Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. 2008. Improving the Performance of Motor-Impaired Users with Automatically-Generated, Ability-Based Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, 1257–1266. doi:10.1145/1357054.1357250

[14] Ziyao He, Syed Fatiul Huq, and Sam Malek. 2025. Enhancing Web Accessibility: Automated Detection of Issues with Generative AI. *Proceedings of the ACM on Software Engineering* 2, FSE (2025), 2264–2287. doi:10.1145/3729371

[15] Jaylin Herskovitz, Andi Xu, Rahaf Alharbi, and Anhong Guo. 2024. ProgramAlly: Creating Custom Visual Access Programs via Multi-Modal End-User Programming. In *The 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. Association for Computing Machinery, 1–15. doi:10.1145/3654777.3676391

[16] Eric Horvitz. 1999. Principles of Mixed-Initiative User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. Association for Computing Machinery, New York, NY, USA, 159–166. doi:10.1145/302979.303030

[17] Syed Fatiul Huq, Ziyao He, Yirui He, and Sam Malek. 2026. Bridging the Gap between Automated Intervention and Actual User Experience: A Mixed-Methods Study on Mobile Accessibility Issues for Screen Reader Users. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 228, 27 pages. doi:10.1145/3772318.3791293

[18] Amy Hurst, Scott E. Hudson, Jennifer Mankoff, and Shari Trewin. 2008. Automatically Detecting Pointing Performance. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI '08)*. ACM, 11–19. doi:10.1145/1378773.1378776

[19] Amy Hurst, Jennifer Mankoff, and Scott E. Hudson. 2008. Understanding Pointing Problems in Real World Computing Environments. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '08)*. Association for Computing Machinery, 43–50. doi:10.1145/1414471.1414481

[20] Shaun K Kane, Jacob O Wobbrock, and Ian E Smith. 2008. Getting off the treadmill: evaluating walking user interfaces for mobile devices in public spaces. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*. 109–118.

[21] Junhan Kong, Mingyuan Zhong, James Fogarty, and Jacob O. Wobbrock. 2024. The Ability-Based Design Mobile Toolkit (ABD-MT): Developer Support for Runtime Interface Adaptation Based on Users' Abilities. *Proceedings of the ACM on Human-Computer Interaction* 8, MHCI, Article 277 (2024), 26 pages. doi:10.1145/3676524

[22] Jonathan Lazar, Aaron Allen, Jason Kleinman, and Chris Malarkey. 2007. What Frustrates Screen Reader Users on the Web: A Study of 100 Blind Users. *International Journal of Human-Computer Interaction* 22, 3 (2007), 247–269. doi:10.1080/10447310709336964

[23] Amanda Li, Jason Wu, and Jeffrey P. Bigham. 2023. Using LLMs to Customize the UI of Webpages. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23 Adjunct)*. ACM, 1–3. doi:10.1145/3586182.3616671

[24] Darren Lunn, Simon Harper, and Sean Bechhofer. 2011. Identifying Behavioral Strategies of Visually Impaired Users to Improve Access to Web Content. *ACM Transactions on Accessible Computing* 3, 4, Article 13 (2011), 35 pages. doi:10.1145/1952388.1952390

[25] Tlamelo Makati, Garreth W. Tigwell, and Kristen Shinohara. 2024. The Promise and Pitfalls of Web Accessibility Overlays for Blind and Low Vision Users. In *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '24)*. ACM, 1–12. doi:10.1145/3663548.3675650

[26] Forough Mehralian, Titus Barik, Jeff Nichols, and Amanda Swearngin. 2024. Automated Code Fix Suggestions for Accessibility Issues in Mobile Apps. https://arxiv.org/abs/2408.03827 arXiv:2408.03827.

[27] Ananya Gubbi Mohanbabu, Rosiana Natalie, Brandon Kim, Anhong Guo, and Amy Pavel. 2026. A11y-CUA Dataset: Characterizing the Accessibility Gap in Computer Use Agents. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 225, 26 pages. doi:10.1145/3772318.3791896

[28] Ananya Gubbi Mohanbabu, Yotam Sechayk, and Amy Pavel. 2025. Task Mode: Dynamic Filtering for Task-Specific Web Navigation using LLMs. In *Proceedings of the 27th International ACM SIGACCESS Conference on Computers and Accessibility*. Association for Computing Machinery, New York, NY, USA, Article 89, 18 pages. doi:10.1145/3663547.3746401

[29] Kyle Montague, Vicki L. Hanson, and Andy Cobley. 2012. Designing for Individuals: Usable Touch-Screen Interaction through Shared User Models. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility*. Association for Computing Machinery, New York, NY, USA, 151–158. doi:10.1145/2384916.2384943

[30] Peya Mowar, Yi-Hao Peng, Jason Wu, Aaron Steinfeld, and Jeffrey P. Bigham. 2025. CodeA11y: Making AI Coding Assistants Useful for Accessible Web Development. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. ACM, 15 pages. doi:10.1145/3706598.3713335

[31] PaulMartz. 2023. Stop reading image file names. https://www.applevis.com/forum/macos-mac-apps/stop-reading-image-file-names. AppleVis forum post, 20 November 2023.

[32] Yi-Hao Peng, Muh-Tarng Lin, Yi Chen, Tzu-Chuan Chen, Pin-Sung Ku, Paul Taele, Chin Guan Lim, and Mike Y. Chen. 2019. PersonalTouch: Improving Touchscreen Usability by Personalizing Accessibility Settings based on Individual User's Touchscreen Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 683, 11 pages. doi:10.1145/3290605.3300913

[33] Letícia Seixas Pereira, Maria Matos, and Carlos Duarte. 2024. Exploring Mobile Device Accessibility: Challenges, Insights, and Recommendations for Evaluation Methodologies. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 964, 17 pages. doi:10.1145/3613904.3642526

[34] Christopher Power, André Freire, Helen Petrie, and David Swallow. 2012. Guidelines Are Only Half of the Story: Accessibility Problems Encountered by Blind Users on the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, 433–442. doi:10.1145/2207676.2207736

[35] r/deaf contributors. 2024. What do you think of the youtube auto generated subtitles? https://www.reddit.com/r/deaf/comments/1d8spkx/what_do_you_think_of_the_youtube_auto_generated/. Reddit discussion, 5 June 2024.

[36] Anne Spencer Ross, Xiaoyi Zhang, James Fogarty, and Jacob O. Wobbrock. 2017. Epidemiology as a Framework for Large-Scale Mobile Application Accessibility Assessment. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*. Association for Computing Machinery, New York, NY, USA, 2–11. doi:10.1145/3132525.3132547

[37] Navid Salehnamadi, Abdulaziz Alshayban, Jun-Wei Lin, Iftekhar Ahmed, Stacy Branham, and Sam Malek. 2021. Latte: Use-Case and Assistive-Service Driven Automated Accessibility Testing Framework for Android. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, 11 pages. doi:10.1145/3411764.3445455

[38] Navid Salehnamadi, Ziyao He, and Sam Malek. 2023. Assistive-Technology Aided Manual Accessibility Testing in Mobile Apps, Powered by Record-and-Replay. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 73, 20 pages. doi:10.1145/3544548.3580679

[39] Navid Salehnamadi, Forough Mehralian, and Sam Malek. 2022. Groundhog: An Automated Accessibility Crawler for Mobile Apps. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22)*. ACM, 13 pages. doi:10.1145/3551349.3556905

[40] Marcia J. Scherer and Ger Craddock. 2002. Matching Person & Technology (MPT) Assessment Process. *Technology and Disability* 14, 3 (2002), 125–131. doi:10.3233/TAD-2002-14308

[41] Omar Shaikh, Shardul Sapkota, Shan Rizvi, Eric Horvitz, Joon Sung Park, Diyi Yang, and Michael S. Bernstein. 2025. Creating General User Models from Computer Use. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25)*. ACM, 1–23. doi:10.1145/3746059.3747722

[42] Amanda Swearngin, Jason Wu, Xiaoyi Zhang, Esteban Gomez, Jen Coughenour, Rachel Stukenborg, Bhavya Garg, Greg Hughes, Adriana Hilliard, Jeffrey P. Bigham, and Jeffrey Nichols. 2024. Towards Automated Accessibility Report Generation for Mobile Apps. *ACM Transactions on Computer-Human Interaction* 31, 4, Article 54 (2024), 44 pages. doi:10.1145/3674967

[43] Maryam Taeb, Amanda Swearngin, Eldon Schoop, Ruijia Cheng, Yue Jiang, and Jeffrey Nichols. 2024. AXNav: Replaying Accessibility Tests from Natural Language. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. ACM, 1–16. doi:10.1145/3613904.3642777

[44] Hironobu Takagi, Shinya Kawanaka, Masatomo Kobayashi, Takashi Itoh, and Chieko Asakawa. 2008. Social Accessibility: Achieving Accessibility through Collaborative Metadata Authoring. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*. Association for Computing Machinery, New York, NY, USA, 193–200. doi:10.1145/1414471.1414507

[45] Shari Trewin and Helen Pain. 1999. A Model of Keyboard Configuration Requirements. *Behaviour & Information Technology* 18, 1 (1999), 27–35. doi:10.1080/014492999119228

[46] Markel Vigo, Justin Brown, and Vivienne Conway. 2013. Benchmarking Web Accessibility Evaluation Tools: Measuring the Harm of Sole Reliance on Automated Tests. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility (W4A '13)*. ACM, 1–10. doi:10.1145/2461121.2461124

[47] Markel Vigo and Simon Harper. 2013. Coping Tactics Employed by Visually Disabled Users on the Web. *International Journal of Human-Computer Studies* 71, 11 (2013), 1013–1025. doi:10.1016/j.ijhcs.2013.08.002

[48] WebAIM. [n. d.]. WAVE Web Accessibility Evaluation Tools. https://wave.webaim.org/. Accessed July 2026.

[49] WebAIM. 2024. Screen Reader User Survey #10 Results. https://webaim.org/projects/screenreadersurvey10/. Accessed July 2026.

[50] WebAIM. 2026. The WebAIM Million: The 2026 report on the accessibility of the top 1,000,000 home pages. https://webaim.org/projects/million/. Accessed September 6, 2026.

[51] Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-Based Design: Concept, Principles and Examples. *ACM Transactions on Accessible Computing* 3, 3, Article 9 (2011), 27 pages. doi:10.1145/1952383.1952384

[52] World Wide Web Consortium. 2023. Web Content Accessibility Guidelines (WCAG) 2.2. https://www.w3.org/TR/WCAG22/. W3C Recommendation, 5 October 2023; revised 12 December 2024.

[53] World Wide Web Consortium. 2026. Technique SCR39: Making Content on Focus or Hover Hoverable, Dismissible, and Persistent. https://www.w3.org/WAI/WCAG22/Techniques/client-side-script/SCR39. Accessed September 11, 2026.

[54] Jason Wu, Gabriel Reyes, Sam C. White, Xiaoyi Zhang, and Jeffrey P. Bigham. 2021. When Can Accessibility Help? An Exploration of Accessibility Feature Recommendation on Mobile Devices. In *Proceedings of the 18th International Web for All Conference (W4A '21)*. ACM, 1–12. doi:10.1145/3430263.3452434

[55] Yaman Yu, Bektur Ryskeldiev, Ayaka Tsutsui, Matthew Gillingham, and Yang Wang. 2025. LLM-Driven Optimization of HTML Structure to Support Screen Reader Navigation. arXiv:2502.18701.

[56] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P. Bigham. 2021. Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. ACM, Article 275, 15 pages. doi:10.1145/3411764.3445186

[57] Xiaoyi Zhang, Anne Spencer Ross, Anat Caspi, James Fogarty, and Jacob O. Wobbrock. 2017. Interaction Proxies for Runtime Repair and Enhancement of Mobile Application Accessibility. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, 6024–6037. doi:10.1145/3025453.3025846

[58] Xiaoyi Zhang, Anne Spencer Ross, and James Fogarty. 2018. Robust Annotation of Mobile Application Interfaces in Methods for Accessibility Repair and Enhancement. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, New York, NY, USA, 609–621. doi:10.1145/3242587.3242616

[59] Mingyuan Zhong, Ruolin Chen, Xia Chen, James Fogarty, and Jacob O. Wobbrock. 2025. ScreenAudit: Detecting Screen Reader Accessibility Errors in Mobile Apps Using Large Language Models. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 19 pages. doi:10.1145/3706598.3713797

[60] Mingyuan Zhong, Xia Chen, Davin Win Kyi, Chen Li, James Fogarty, and Jacob O. Wobbrock. 2026. TaskAudit: Detecting Functiona11ity Errors in Mobile Apps via Agentic Task Execution. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*. ACM, 20 pages. doi:10.1145/3772318.3791415

# A Study Tasks

Table 4. Study tasks attempted.

| Participants | Application | Activity |
|---|---|---|
| P1, P3, P7, P10 | Craigslist | Find classified listings. |
| P1, P3, P7, P10 | CAPTCHA demo | Complete an image CAPTCHA. |
| P1, P3, P7, P10 | Local arts center | Inspect venue photographs. |
| P1, P3, P7, P10 | When2Meet | Mark availability. |
| P1, P3, P7, P10 | Kill Crew | Check clothing size availability. |
| P1 | RYSE | Modify cart items. |
| P7, P10 | Fly Southern | Explore flight booking. |
| P2, P4, P5, P6 | Uber Eats | Select food, flavors, and quantities. |
| P2, P4, P5, P6 | Google Flights | Select dates and compare flights. |
| P2, P4, P5, P6 | H&M | Add clothing and check confirmation. |
| P2, P4, P5, P6 | Restaurant | Read the menu. |
| P2, P4, P5, P6 | RYSE | Change quantities and remove items. |
| P2, P4, P5 | Amazon | Browse hover menus. |
| P8, P9 | When2Meet | Create an event and mark availability. |
| P8, P9 | Booking.com | Search, filter, and book accommodation. |
| P8 | Pages | Type and format a notice. |
| P8 | Facebook | Compose a post. |
| P8 | University news site | Read an article. |
| P8 | macOS | Use system shortcuts. |
| P8 | Gmail | Delete and undo an email. |
| P9 | Pages and Google Slides | Create a notice and illustrated slide. |
| P9 | Qualtrics | Complete varied survey controls. |

# B Evaluation Details

## B.1 Reproduced-Barrier Dataset

Table 5. GAM-Eval-RB coverage by access group.

| Access group | Categories | Scenarios | Recordings |
|---|---|---|---|
| Blind | 5 | 12 | 23 |
| Low vision | 6 | 12 | 19 |
| Motor | 5 | 5 | 10 |
| DHH | 2 | 5 | 10 |
| Total | 18 | 34 | 62 |

## B.2 Input Counts

Table 6. Processed segments by input configuration.

| Measure | GAM-Eval-RB | GAM-Eval-US |
|---|---|---|
| Full-observation segments | 227 | 644 |
| Static-input segments | 170 | 481 |

### B.3 Detailed Counts and Ratings

Table 7. Recalled/documented barriers by group.

| Dataset | Group | Full observation | Static input |
|---|---|---|---|
| GAM-Eval-RB | Blind | 23/23 | 19/23 |
| GAM-Eval-RB | Low vision | 19/19 | 11/19 |
| GAM-Eval-RB | Motor | 9/12 | 3/12 |
| GAM-Eval-RB | DHH | 10/10 | 6/10 |
| GAM-Eval-RB | All | 61/64 | 39/64 |
| GAM-Eval-US | Blind | 28/39 | 23/39 |
| GAM-Eval-US | Low vision | 29/45 | 9/45 |
| GAM-Eval-US | Motor | 6/20 | 1/20 |
| GAM-Eval-US | All | 63/104 | 33/104 |

Table 8. Recalled/documented barriers by participant.

| Participant | Group | Full observation | Static input |
|---|---|---|---|
| P1 | Blind | 5/8 | 5/8 |
| P2 | Low vision | 7/11 | 2/11 |
| P3 | Blind | 5/6 | 5/6 |
| P4 | Low vision | 8/12 | 4/12 |
| P5 | Low vision | 9/15 | 3/15 |
| P6 | Low vision | 5/7 | 0/7 |
| P7 | Blind | 12/16 | 8/16 |
| P8 | Motor | 1/7 | 0/7 |
| P9 | Motor | 5/13 | 1/13 |
| P10 | Blind | 6/9 | 5/9 |
| Total | | 63/104 | 33/104 |

Table 9. Accuracy of unmatched barriers and interaction patterns.

| Dataset | Proposition type | Reviewed | Accurate | Inaccurate |
|---|---|---|---|---|
| GAM-Eval-RB | Accessibility barrier | 30 | 30 | 0 |
| GAM-Eval-RB | Interaction pattern | 69 | 65 | 4 |
| GAM-Eval-US | Accessibility barrier | 76 | 76 | 0 |
| GAM-Eval-US | Interaction pattern | 47 | 44 | 3 |
| All | | 222 | 215 | 7 |

Table 10. Repair rating counts and sensitivity.

| Measure | Value |
|---|---|
| Ratings / participants | 30 / 9 |
| Live / previously generated / uncertain provenance | 13 / 14 / 3 |
| Adjacent-value responses | 3 |
| Mean using lower / upper endpoints | 5.73 / 5.83 |
| Single-number responses | 27 |
| Mean for single-number responses | 5.78 |
| Excluded response with unclear scale interpretation | 1 |