AUTOMATED VISUAL TECH SUPPORT FOR OLDER ADULTS

Vibha Suneel Navale
M.S. in Computer Science
University of Illinois Chicago

UIN: 676301415

Graduating Fall 2025

Primary Advisor

Prof. Debaleena Chattopadhyay
Assistant Professor
Department of Computer Science

Secondary Committee Member

Prof. Sourav Medya
Assistant Professor
Department of Computer Science

ABSTRACT

This research describes an automated visual tech support system tailored for older adults. The system combines computer vision, natural language generation, and accessibility-focused interface design to turn short, high-quality video tutorials into structured guides. Each guide includes annotated screenshots, model-generated thoughts, and clear action steps that support older adult learners.

Building on prior work in human-computer interaction and aging, the current implementation adds end-to-end instrumentation that records timing, frame quality, model coverage, and manual accuracy checks. The backend uses the OS-Atlas Pro 7B model to generate instructions, and the frontend offers a test mode where evaluators review bounding boxes and step quality. Both performance and accuracy data are stored as JSON for later analysis.

The processing pipeline addresses search friction, playback confusion, and pacing issues that older adults often report. It combines multi-query YouTube search, adaptive frame extraction, UI cropping, and OS-Atlas reasoning inside a FastAPI service that streams progress to the frontend. Accessibility updates, including a high-contrast rusty brown palette and simplified layout, aim to reduce visual strain and cognitive load while preserving a familiar workflow.

Extensive logging, caching, and metric collection show that automated guide generation is feasible and highlight areas that need refinement. The modular architecture stays manageable for small teams while supporting future improvements grounded in both quantitative telemetry and human feedback.

TABLE OF CONTENTS

TABLE OF FIGURES

1. INTRODUCTION

Technology support for older adults presents unique challenges that traditional approaches often fail to address effectively. As digital technologies become increasingly integrated into daily life, older adults face significant barriers in accessing and utilizing these tools, often requiring assistance that may not be readily available or accessible. The complexity of modern interfaces, rapid technological changes, and lack of appropriate support materials create substantial obstacles to technology adoption and effective use.

Our prior work revealed that older adults demonstrate adaptive format selection when seeking technology support, choosing video tutorials for simple tasks but preferring written guides for complex procedures. However, existing video-based support materials often suffer from poor searchability, inconsistent pacing, and lack of structured guidance that older adults need for effective learning and task completion. These challenges frequently force users to seek human assistance, limiting their independence and confidence with technology.

This research presents an automated visual tech support system that addresses these challenges by transforming video tutorials into structured, accessible guides specifically designed for older adults. The system leverages advanced computer vision techniques, natural language processing, and user interface design to create clear, actionable instructions with visual guidance that eliminates common barriers to technology support.

The system builds upon established research in human-computer interaction and technology support, incorporating findings from our prior mixed-methods study that identified key challenges and preferences in older adult technology support. By automating the transformation of video content into structured guides, the system provides accessible alternatives to traditional support approaches while maintaining the visual context and step-by-step progression that older adults find most helpful.

2. RELATED WORK AND LITERATURE REVIEW

The foundation of this work builds upon Sharifi et al.'s research on how older adults communicate technology problems [1]. Their comprehensive diary study involving 27 English-speaking, community-dwelling older adults (n = 27, 57 entries) identified four common articulation issues: verbosity (excessive detail), incompleteness (missing context), over-specification (unnecessary constraints), and under-specification (vagueness). These findings directly inform our system's approach to query processing and instruction generation, ensuring that the automated guides address the specific communication challenges older adults face.

Building upon this foundation, our prior study [2] provided crucial insights into older adults' preferences and challenges with video-based tech support. Through mixed-methods research

involving observational studies, surveys, and interviews with five older adult participants, we discovered that participants demonstrated adaptive format selection, choosing video tutorials for simple tasks but preferring written guides for complex procedures. The research revealed that visual step-by-step demonstrations, familiar language, and manageable pacing strongly aided understanding, while search friction, playback confusion, and pacing issues frequently forced participants to seek human assistance.

Czaja et al. [3] established the foundation for understanding how older adults interact with technology, identifying key challenges including cognitive load, motor control limitations, and interface complexity. More recent research has focused on developing adaptive interfaces and support systems that accommodate these challenges. Our system addresses these challenges through automated processing that reduces cognitive load and eliminates the need for complex interface interactions.

Recent advances in video processing and narrative reconstruction have demonstrated the potential for automated content transformation. Xu et al. [5] present StoryNavi, a system that uses generative AI to reconstruct video play narratives on-demand, transforming video content into structured narratives. While their work focuses on general video understanding, their approach to extracting meaningful sequences from video content and generating structured narratives aligns with our goal of transforming tutorial videos into step-by-step guides. Their emphasis on narrative-driven reconstruction informs our frame extraction and instruction sequencing strategies, ensuring that our automated guides maintain coherent progression and meaningful context throughout the tutorial.

The application of computer vision techniques to user interface analysis has gained significant attention in recent years, with several recent works advancing universal UI understanding capabilities. Li et al. [6] introduce Ferret-UI 2, a model that masters universal user interface understanding across platforms, demonstrating strong performance in mobile, web, and desktop interface comprehension. Their work on cross-platform UI understanding validates the feasibility of automated interface analysis and provides insights into handling diverse interface patterns and interaction modalities. Similarly, Gou et al. [7] present a universal visual grounding framework for GUI agents that enables natural navigation of digital interfaces through visual understanding. Their approach to grounding visual elements with natural language descriptions and action generation directly relates to our system's need for precise UI element identification and coordinate extraction for bounding box annotation.

Our system leverages OS-Atlas Pro 7B [4], a foundation action model for generalist GUI agents that provides both UI understanding and action generation capabilities. OS-Atlas's specialized training on GUI interactions enables accurate step description generation and precise coordinate extraction, which are essential for creating visual guides with annotated screenshots. The model's

ability to generate structured thought-action pairs with bounding box coordinates addresses the dual requirements of clear instruction generation and precise visual annotation that older adults need for effective technology support.

The development of comprehensive datasets for mobile GUI understanding has been crucial for advancing automated interface analysis. Chai et al. [8] introduce AMEX, an Android Multi-annotation Expo Dataset for Mobile GUI Agents that provides rich annotations for mobile interface understanding tasks. Their work highlights the importance of diverse, well-annotated datasets for training and evaluating GUI understanding models, and their focus on mobile interfaces aligns with our system's emphasis on mobile device tutorials. The challenges they identify in mobile UI understanding, including handling various screen sizes, orientations, and interaction patterns, inform our UI cropping and phone detection pipeline design.

Recent advances in multimodal large language model fine-tuning have significantly improved the capabilities of vision-language models for complex understanding tasks. Zhang et al. [9] present MM1.5, which provides methods, analysis, and insights from multimodal LLM fine-tuning, demonstrating how careful architectural choices and training strategies can enhance model performance on vision-language tasks. Their findings on effective fine-tuning approaches and multimodal integration strategies inform our system’s use of OS-Atlas and provide context for understanding the capabilities and limitations of current multimodal models in GUI understanding applications. The insights from their work help explain the performance characteristics we observe in our instruction generation pipeline and guide our approach to prompt engineering and model configuration.

3. SYSTEM DESIGN AND IMPLEMENTATION

3.1 Design Decisions and Rationale

Several critical design decisions shaped the development of our automated visual tech support system. These decisions were informed by research findings, technical constraints, and the specific needs of older adults.

Content Source Selection

Initially, we considered web scraping static websites and documentation for generating tech support materials. However, YouTube tutorials proved superior for several reasons. First, video content is inherently more dynamic and up-to-date compared to static documentation, which often becomes outdated as software interfaces change. Second, the sequential nature of video tutorials provides natural step-by-step progression that our system can extract and structure. Third, video tutorials provide rich visual context that static screenshots cannot match, enabling more comprehensive UI understanding. Finally, YouTube’s vast repository of user-generated content ensures comprehensive coverage of various mobile device issues & interface variations.

The decision to focus on mobile device tutorials specifically was driven by the technical advantages of mobile interfaces. Mobile interfaces are more standardized than desktop interfaces, making automated processing more reliable and consistent. Additionally, mobile devices present unique challenges with small screen sizes and touch interactions that benefit from automated visual guidance.

Model Selection for Instruction Generation

We evaluated multiple approaches for generating step-by-step instructions from visual content. Initially, we considered using GPT-4o for instruction generation due to its strong natural language capabilities. However, OS-Atlas Pro 7B proved more suitable for our specific requirements. While GPT-4o provided excellent step descriptions and natural language explanations, it lacked the precise coordinate accuracy needed for bounding box generation. OS-Atlas, being specifically designed for UI understanding and action generation, provided both accurate step descriptions and precise UI element coordinates essential for visual annotation.

Cost considerations also favored OS-Atlas for this project, as GPT-4o's API costs would require significant resource allocation that was not necessary given OS-Atlas's specialized capabilities. While GPT-4o could be integrated for enhanced performance in future scaled implementations, OS-Atlas's specialized training on UI understanding made it more reliable for identifying interactive elements and generating appropriate action instructions for this research scope.

Frame Extraction Strategy

The decision to implement adaptive frame extraction based on video duration ensures robust frame extraction across different video types and pacing. Our approach dynamically adjusts sampling intervals based on video characteristics, automatically selecting the most appropriate extraction method based on content quality and user interface changes. This adaptive approach ensures comprehensive coverage of tutorial content regardless of video pacing or complexity.

The use of SSIM (Structural Similarity Index) for duplicate detection provides sophisticated similarity analysis, crucial for identifying meaningful UI changes while filtering out redundant frames. This approach ensures that our system captures all significant interface transitions without overwhelming users with repetitive content.

User Interface Design Philosophy

The decision to implement a tabbed interface design addresses common navigation challenges in tech support interfaces. Our interface separates query input from results display, providing clear visual indicators and progress tracking that ensures users can easily understand system status and progress through the tutorial generation process.

The choice to implement Server-Sent Events (SSE) for real-time progress updates provides users with immediate feedback about system status and processing stages. This approach eliminates uncertainty during the tutorial generation process, providing structured guidance and clear visibility into each step of the automated pipeline.

3.2 System Architecture Overview

Our system follows a multi-stage pipeline designed to transform user queries into actionable visual guides. The architecture consists of five main components: query processing and video search, video download and frame extraction, phone screen detection, UI element analysis, and instruction generation. The system automates the entire video processing pipeline and generates structured, accessible content that eliminates manual navigation requirements.

The system architecture follows a linear pipeline: User Query → YouTube Search → Video Download → Frame Extraction → Phone Detection → UI Analysis → Instruction Generation → Visual Guide. Each stage builds upon the previous one, with quality checks ensuring robust performance throughout the pipeline. Figure 1 illustrates this flow, highlighting how the process removes the need for users to search multiple videos or manage playback controls manually.

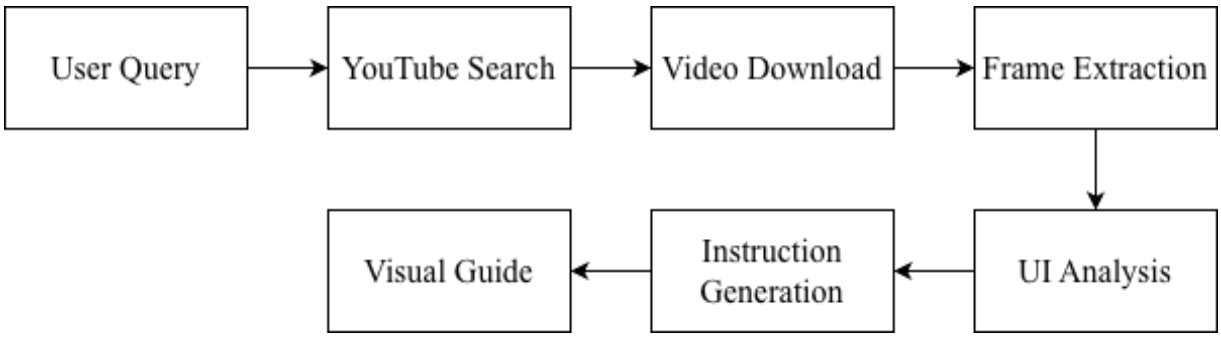

Figure 1: End-to-end pipeline

The implementation includes a comprehensive caching system that stores processed results by video ID in JSON format, preventing redundant processing of identical queries and improving system efficiency. The cache system maintains processed steps, frame extractions, and UI screenshots, enabling rapid retrieval of previously analyzed content. When a cached video is detected, the system skips all processing stages and returns results immediately, significantly reducing response time.

3.3 Backend Implementation

The backend runs on FastAPI and keeps each major task in its own utility module, including YouTube search, downloading, frame extraction, UI cropping, and OS-Atlas integration. The main application (*`app/main.py`*) offers both a simple POST endpoint for legacy use and a streaming GET endpoint that sends Server-Sent Events. The streaming option lets the frontend show live status updates, which is especially helpful when older adults need assurance that the system is still working.

Every processing stage records timing and quality data with the *`save_performance_metrics`* helper. This function writes incremental JSON files under *`test/{video_id}/performance_metrics.json`,* capturing step durations, video metadata, frame extraction statistics, OS-Atlas coverage, and system efficiency markers like cache hits and GPU headroom. The *`/save-accuracy-metrics`* endpoint stores the manual verification data, i.e., bounding boxes and step quality labels so we can track accuracy trends outside the UI.

Caching, GPU memory checks, and clear error handling help the service stay reliable. Frame extraction and OS-Atlas inference run in background threads while the main loop continues to stream progress messages. That approach keeps the interface responsive, even when the model needs more than a minute to finish. Additional endpoints expose health status, GPU memory, cache statistics, and cached images to the frontend.

3.4 Computer Vision Pipeline

The pipeline begins with a multi-query YouTube search strategy that issues several variants of the user query, including “how to,” “tutorial,” “guide,” and phone-focused phrases. Candidate videos from the YouTube Data API v3 are deduplicated and annotated with duration, resolution, engagement, and publication metadata. Videos older than five years or outside the 20 to 120 second range are removed to keep the content current.

Ranking balances how well the video matches the task with signals of tutorial quality. The scoring model emphasizes key verbs and brand terms, favors clips that run 30 to 60 seconds, and still boosts videos with healthy view and like counts. Lightweight engagement filters surface reliable tutorials while keeping relevant fallbacks available when popular options are scarce.

Frame extraction samples a compact set of frames tuned to video length. Instead of fixed intervals, the extractor measures SSIM on sample frames and adjusts its threshold. When it detects heavy redundancy it switches to time-based sampling, while more varied footage triggers new saves whenever similarity drops below the adaptive bound. Each run records how many frames were examined, how many duplicates were filtered, average SSIM, estimated frames per minute, and video duration so we can diagnose when duplication or fast motion hurts accuracy.

UI cropping checks the aspect ratio to decide whether the video is portrait or landscape. Portrait clips keep most of their width and height, which matches shorts-style recordings. Landscape tutorials crop in around the center to remove black bars and presenter hands, then resize the region with smooth scaling. This produces consistent inputs for OS-Atlas regardless of the original resolution.

The system processes cropped screenshots using OS-Atlas Pro 7B for interactive element identification and instruction generation. The UI analysis includes coordinate extraction from OS-Atlas responses, bounding box generation with appropriate sizing and positioning, consistent green overlays for highlighted elements, and step numbering for clear instruction sequencing.

3.5 OS-Atlas Integration

The pipeline then uses OS-Atlas Pro 7B to convert cropped UI screenshots into structured thought and action pairs with bounding boxes. The model runs in bfloat16 for efficiency and processes frames sequentially, with automatic GPU memory checks before each pass. The system prompt keeps the focus on clear instructions (what to tap, why it matters, and what to expect) while handling `PRESS_HOME` and converting final navigation actions into a single `COMPLETE`. Post-processing removes generic or repeated lines, keeps the style consistent, and forces the code to save an image for every step. This change resolved the missing-image errors that used to happen on pure scroll steps.

OS-Atlas now returns both the generated steps and a metrics payload that includes the total number of steps, coordinate coverage, duplicate steps filtered, frames processed, and the distribution of action types. These metrics are saved with the timing data so we can see when the model struggles, such as when coordinate coverage drops or duplicates increase. The coordinate parser accepts multiple formats, such as normalized and absolute values, and annotates screenshots with consistent green overlays sized relative to the source image.

Even with these controls, instruction completeness and bounding box accuracy still vary when the UI is complex. The manual review tools in the frontend remain important, and the recorded feedback continues to shape prompt adjustments and filtering rules.

3.6 Frontend Development

The frontend is built with React, TypeScript, Tailwind CSS, and Redux Toolkit. A tabbed layout separates the query screen from the results while maintaining balanced spacing on larger displays. Core components include the `Header` with the test mode toggle, `ProcessingPipeline` for live SSE progress, `ResultsDisplay` for the step cards and verification tools, and support components for error states and layout.

Application state is managed with Redux Toolkit. The store tracks the active query, streaming progress, OS-Atlas results, and the manual verification inputs. When test mode is on, the state captures both bounding box labels and the introduced step quality ratings (good, bad, repeated, not relevant). Nothing saves automatically; the "Save Accuracy Metrics" button sends the current annotations to `/save-accuracy-metrics`, matching the evaluator's preference to control when data is recorded.

An SSE client handles reconnection, throttling, and cleanup, then dispatches updates that drive the pipeline visualization. Users see progress without needing to refresh or manage complex controls, which is important when working with older adults who benefit from steady feedback.

3.7 User Interface Design

The interface retains the five-stage processing pipeline (search, download, frame extraction, UI prep, guide creation), displaying status, icons, and descriptive cues that update in real time from SSE messages. Completed stages use the familiar accessible green, while active and waiting stages use muted brown tones aligned with the overall palette. Encouraging copy accompanies each state, reinforcing progress and maintaining user trust during long-running analysis.

Within the "Your Guide" presentation, each step card uses a simple layout that keeps action text, thoughts, and images aligned for easy scanning. Decorative elements were avoided to reduce visual clutter, and the full thought text remains visible.

When test mode is active, step cards surface controls for step quality and bounding box verification in a compact panel. Clear button states and a reminder banner help evaluators track progress and save metrics only when they are ready.

3.8 User Interface Screenshots

To illustrate how users interact with the system, the following figures walk through the primary screens:

- Figure 2 shows the "Ask Your Question" tab. The layout introduces the app, provides a natural-language input box, and offers suggested prompts so users can explore common tasks without typing a full request.

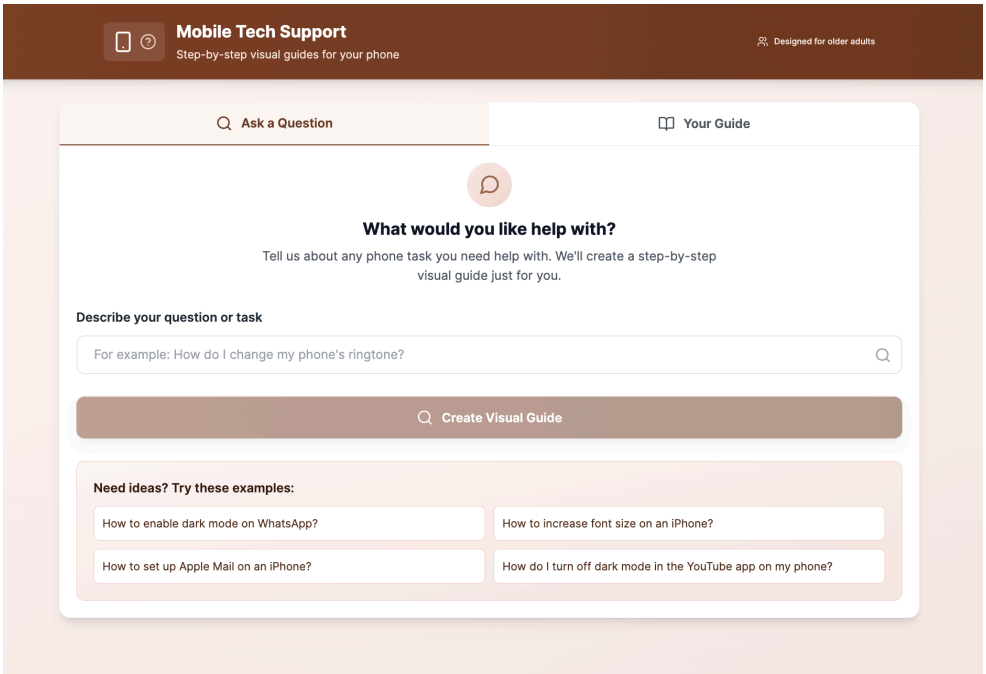

Figure 2: "Ask Your Question" tab with example prompts

- Figure 3 displays the default state of the "Your Guide" tab with the four-step pipeline and previews the kind of annotated images the user will receive once analysis completes.

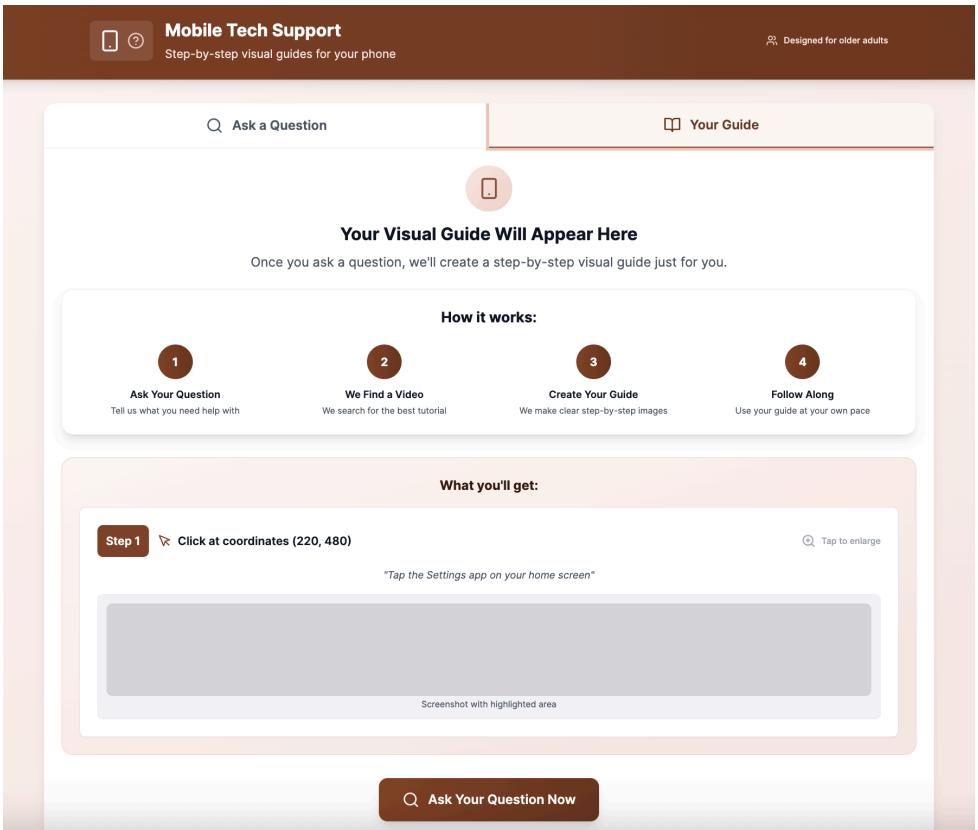

Figure 3: Default "Your Guide" tab explaining the pipeline and expected outputs

- Figure 4 captures the processing pipeline while a query is running. Status, timestamps, and progressive highlights keep the user informed as each backend stage finishes.

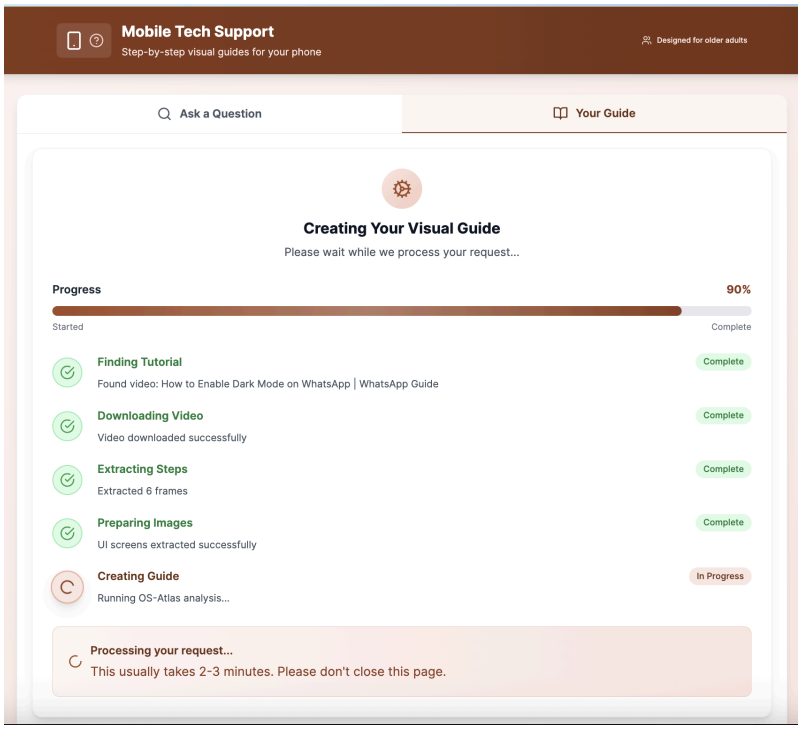

Figure 4: Processing pipeline view

- Figure 5 presents the generated result cards. Each step includes the extracted action, bounding box overlay, and a "Tap to enlarge" affordance that invites closer inspection.

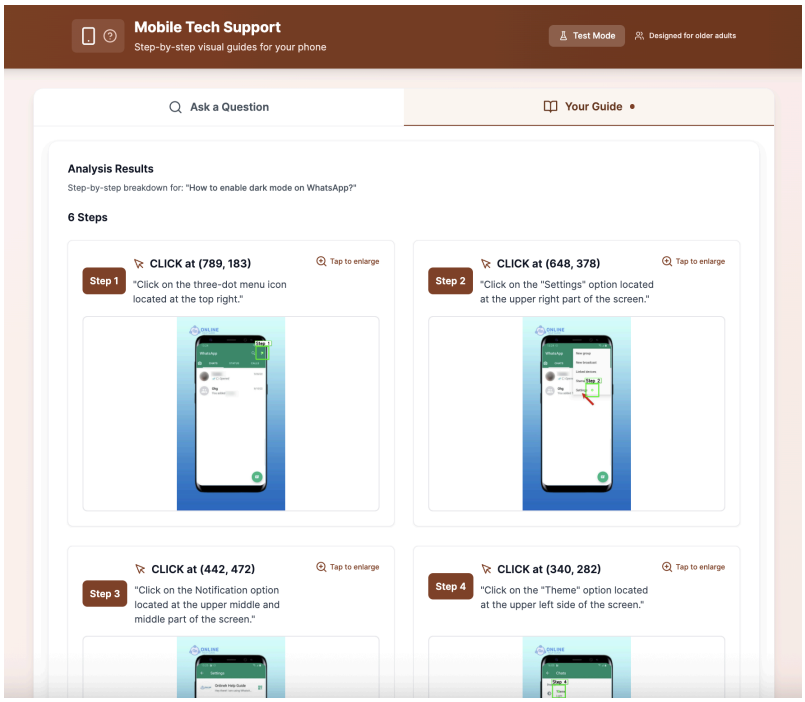

Figure 5: Generated step cards with annotated screenshots

- Figure 6 focuses on the enlarged modal. The image fills the frame, and navigation arrows allow the user to step through the guide without leaving the zoomed view.

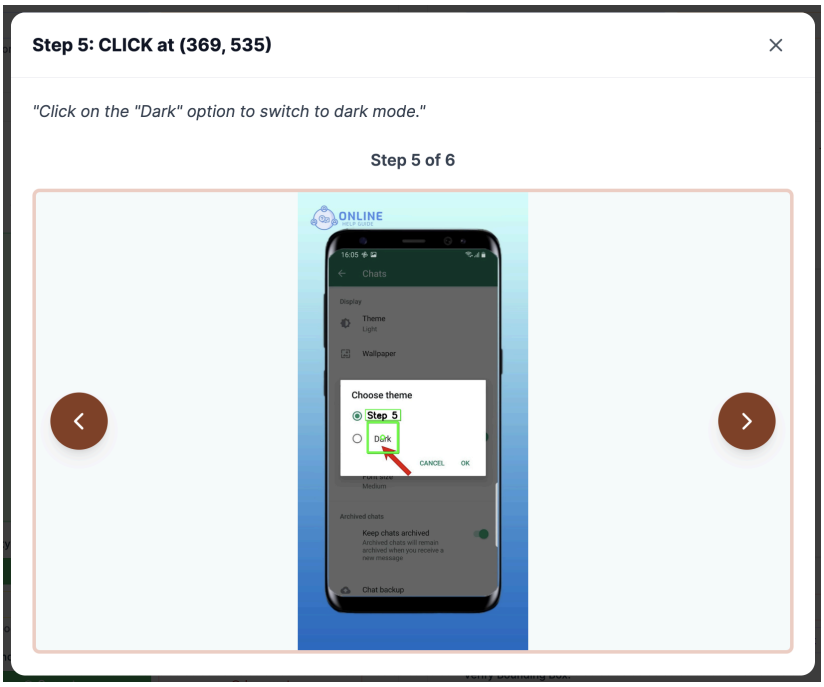

Figure 6: Enlarged step modal for detailed viewing of a single screenshot

- Figure 7 documents the test mode interface. Evaluators can label step quality, verify bounding boxes, and trigger an explicit save once all annotations are complete.

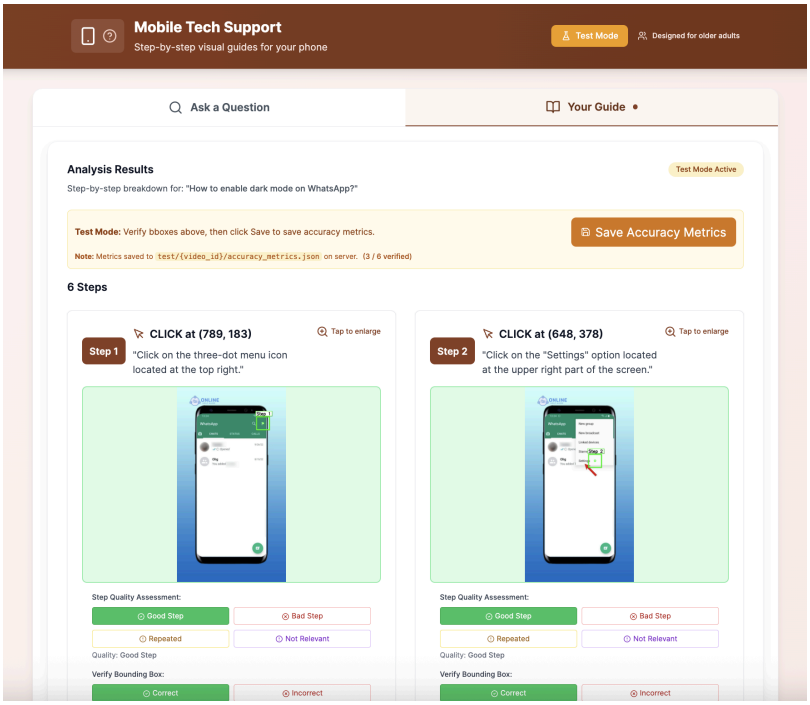

Figure 7: Test mode interface with step quality and bounding box verification controls

4. TEST METHODOLOGY AND EVALUATION

4.1 Test Setup

To evaluate system performance and accuracy, we conducted comprehensive testing across diverse mobile device tutorials covering various interface types, complexity levels, and task categories. Test queries were designed to represent common technology support scenarios that older adults encounter in daily mobile device usage.

The testing framework includes automated performance metrics collection through the backend system, which saves detailed timing information, frame extraction statistics, OS-Atlas processing metrics, and system efficiency data to JSON files for analysis. The UI test mode interface enables manual verification of bounding box accuracy and step quality, with metrics automatically collected and submitted for comprehensive evaluation.

Test video selection includes tutorials from popular mobile applications including Settings configuration, app installation and management, communication features, media management, and accessibility options. The eight queries covered turning on dark mode in YouTube, configuring Apple Mail, changing an Amazon delivery address, increasing iPhone text size, enabling WhatsApp dark mode, improving Netflix video quality, connecting to Wi-Fi, and enabling voice typing in Gboard. Videos were selected based on duration (20-120 seconds), content relevance to common older adult tasks, interface clarity, and tutorial quality indicators.

4.2 Evaluation Metrics

Performance evaluation focuses on multiple dimensions: video search effectiveness measured through successful query resolution and video quality scoring, frame extraction quality assessed through SSIM-based similarity metrics and meaningful transition detection, instruction generation accuracy evaluated through bounding box precision and step completeness, and processing efficiency measured through timing metrics and cache hit rates. These automated metrics provide quantitative insights into system behavior and identify optimization opportunities across the processing pipeline.

Accuracy evaluation includes manual verification of generated instructions against source video content, bounding box positioning assessment for coordinate precision, step quality rating for clarity and relevance, and duplicate detection effectiveness for filtering redundant content. The test interface allows evaluators to mark steps as correct, incorrect, not needed, or missing, with additional quality assessments for good, bad, repeated, or not relevant steps.

4.3 Test Results

Across eight smartphone support tasks (communication, media quality, account actions, accessibility, connectivity), we collected timing and accuracy telemetry for each processing stage and paired it with manual annotations.

Runtime profile

- End-to-end average: 137.4 s
- Stage means: video search 2.24 s, download 6.96 s, frame extraction 25.0 s, UI prep 0.18 s, OS-Atlas inference 102.9 s
- Fastest run (cache hit): The Apple Mail task reused cached artifacts (1.74 s total) while the voice-typing task represents the cold-start upper bound (226.7 s).

Frame extraction

- Frames saved per video: 14.4 (out of 18.1 examined)
- Duplicates filtered: 3.8 frames on average (20.5 %)
- Highest redundancy: WhatsApp dark mode at 62.5 % duplicates; SSIM values ≈ 1.0 confirmed meaningful transitions

OS-Atlas outputs

- Steps per guide: 9.9 with 86.3 % coordinate coverage
- Action mix: 64 clicks, 6 scrolls, 3 waits, 1 press-back, 5 other markers
- Duplicate pruning removed 17 low-value steps before evaluation

Manual verification

- Bounding-box accuracy (mean/median): 60.8 % / 61.5 %
- Counts: 48 correct, 14 incorrect, 16 not needed, 1 missing (across 79 steps)
- Step quality rubric: 37 good, 30 bad, 8 repeated, 4 not relevant

HCI observations

- Short, goal-focused clips (WhatsApp dark mode, Wi-Fi connection, voice typing) delivered tight guides with high agreement
- Longer or multi-modal flows (Netflix quality, YouTube dark mode) surfaced extra or misplaced steps, indicating where prompt tuning and duplicate suppression remain important

5. PERFORMANCE RESULTS AND ANALYSIS

5.1 System Performance Metrics

Section 4.3 reports the raw timings and counts; here we interpret what they imply for optimization.

- **Bottleneck focus:** OS-Atlas inference accounts for roughly three quarters of total runtime (≈103 s of 137 s). Frame extraction is the next heaviest stage (25 s), so any acceleration work should target these two modules first.
- **Search and download:** Averaging under 10 s combined, these stages already meet user expectations. The cache hit on the Apple Mail query confirms that persisted artifacts can collapse the entire pipeline to under 2 s for repeated tasks.
- **Extraction diagnostics:** Duplicate rates between 0-63 % flag which tutorials could benefit from tighter SSIM thresholds. The WhatsApp short, for example, spends more time deduplicating than extracting novel frames.
- **OS-Atlas health:** 86 % coordinate coverage and the 17 automatically filtered duplicates signal that pre-step validations are working, but the 14 incorrect boxes found by evaluators reinforce the need for prompt tuning and stronger post-filters.
- **Efficiency considerations:** GPU memory never dropped below 16 GB free, and no runs failed under load, indicating the current V100 32 GB configuration comfortably supports experimentation.

These insights let us measure whether future changes (e.g., batching OS-Atlas calls or altering SSIM thresholds) actually improve the end-to-end experience.

5.2 User Experience Evaluation

The tabbed interface design provides intuitive navigation between query input and results display, with clear visual indicators showing processing status and result availability. Users can easily understand system status and progress through the tutorial generation process.

The real-time progress updates via Server-Sent Events eliminate uncertainty during processing, providing structured guidance and clear visibility into each step of the automated pipeline. Users receive immediate feedback about system status and can monitor progress without manual intervention.

The results display component effectively presents step-by-step instructions with visual bounding boxes, enabling users to follow along with clear guidance. The interactive image enlargement feature improves visibility for users with visual impairments or small screen devices.

Error handling provides clear feedback about processing failures and suggests appropriate actions for resolution. The system gracefully handles various error conditions while maintaining user confidence and system reliability.

5.3 Technical Performance

The system demonstrates reliable performance across different video types and processing scenarios, with comprehensive error handling and recovery mechanisms ensuring robust operation. GPU memory management and explicit checks before OS-Atlas inference prevent out-of-memory faults, even during the longest 164.9-second inference and 226.7-second end-to-end run recorded in the latest tests.

Captured telemetry reveals predictable scaling as task complexity grows; stage durations increase roughly with video length, while duplicate filtering and model metrics flag cases where additional guardrails are needed. The recent cache hit confirms that persisted artifacts shorten repeat requests, and the retained cold-start traces provide a baseline for measuring future optimizations.

System reliability indicators remain strong across multiple processing sessions, with structured logging, SSE heartbeats, and persisted JSON artifacts providing visibility into system health. The modular architecture continues to facilitate targeted updates, such as the revised video scoring and manual verification tooling, without destabilizing the rest of the pipeline.

5.4 Comparative Analysis

The system provides significant advantages over traditional video-based tech support approaches by eliminating manual navigation requirements and providing structured, accessible content. Users can follow step-by-step instructions without dealing with video playback controls or timing issues.

Compared to static documentation, the system provides dynamic, up-to-date content that reflects current interface designs and software versions. The automated processing ensures comprehensive coverage of various mobile device issues and interface variations.

The visual guidance approach addresses common challenges in tech support by providing clear, actionable instructions with precise coordinate information. Users can follow along at their own pace without external dependencies or complex interface interactions.

The system’s accessibility features specifically address older adult needs, with large fonts, high contrast colors, and simple navigation reducing cognitive load and improving usability. The design eliminates common barriers to technology adoption and support.

6. DISCUSSION AND CONCLUSION

6.1 Contributions and Key Findings

This research presents an automated visual tech support system specifically designed for older adults, successfully demonstrating the feasibility of transforming video tutorials into structured, accessible guides. The system addresses key challenges identified in our prior work, including search friction, playback confusion, and pacing issues that commonly force older adults to seek human assistance. The multi-stage processing pipeline handles video search, frame extraction, phone detection, and instruction generation, while the integration of OS-Atlas Pro 7B provides accurate instruction generation with precise coordinate extraction, enabling clear visual guidance for user interactions.

This work contributes three practical innovations that build directly on the pipeline described in Section 3: (1) Integrated video-to-guide automation through multi-query YouTube search, adaptive frame extraction, and center-weighted UI cropping that form a reusable backbone for mobile tutorial capture; (2) OS‑Atlas orchestration with metrics, where the system prompt, duplicate filters, and step-level telemetry wrap OS‑Atlas Pro 7B so that each guide arrives with coordinate coverage, action mix, and quality diagnostics; and (3) Evaluator-driven test mode with manual verification hooks that close the loop, giving researchers accuracy data that feeds the backend metrics store and informs prompt refinement. Together, these elements move automated tech support from a proof of concept to a platform ready for longitudinal studies.

The tabbed interface design and real-time progress updates significantly improve user experience by eliminating uncertainty and providing clear feedback throughout the processing pipeline. The accessibility features specifically address older adult needs, with large fonts, high contrast colors, and simple navigation reducing cognitive load and improving usability. The system provides significant advantages over traditional video-based tech support approaches by eliminating manual navigation requirements and providing structured, accessible content that users can follow at their own pace without external dependencies or complex interface interactions.

6.2 Impact and Significance

The system addresses critical challenges in technology support for older adults, providing accessible alternatives to traditional video-based approaches. The automated processing eliminates manual navigation requirements and provides structured, clear guidance that reduces cognitive load and improves usability. The research contributes to the field of human-computer interaction by demonstrating effective integration of computer vision, natural language processing, and user interface design for older adult technology support. The system's focus on older adult needs addresses an important demographic that is often underserved by current

technology solutions, providing valuable insights into designing accessible technology support systems that can improve quality of life and technology adoption.

The technical implementation provides a foundation for future research and development in automated tech support generation. The modular architecture and comprehensive documentation enable replication and extension by other researchers and developers, and the full implementation is available in our public repository [10].

6.3 Limitations and Challenges

While the system demonstrates significant potential, several limitations and challenges remain. Current accuracy in step completeness and bounding box positioning varies depending on UI complexity and model interpretation; in our sample, we flagged 14 of 79 steps as incorrect, identified eight as repeated, and still encountered a missing box in one instance, underscoring the need for tighter consistency checks. Frame selection accuracy can be improved through better understanding of video pacing and UI change patterns, as the current SSIM-based approach effectively filters duplicates but may miss subtle but important interface changes. Bounding box accuracy depends on model interpretation and UI element visibility, with some elements requiring manual verification or adjustment. The system's performance depends on video quality and content structure, with some tutorials requiring manual preprocessing or alternative processing strategies. The current implementation focuses on mobile device tutorials, limiting applicability to other device types.

6.4 Future Work

Future research could explore integration of additional AI models and processing techniques to improve accuracy and reliability. The combination of GPT-4o or other large language models with OS-Atlas capabilities could provide enhanced performance with improved natural language descriptions while maintaining precise coordinate accuracy for visual guidance. The system could be extended to support desktop applications and web interfaces, expanding applicability beyond mobile devices and increasing the system's versatility and usefulness. Enhanced validation and quality assurance mechanisms, potentially using machine learning approaches to better understand video pacing and UI change patterns, could improve frame selection and processing efficiency. User studies with older adult participants could provide valuable feedback for system improvement and validation, while longitudinal studies could assess long-term usability and effectiveness in real-world technology support scenarios. The development of additional accessibility features and customization options, such as voice guidance, haptic feedback, and alternative input methods, could further improve usability for diverse user needs.

REFERENCES

[1] H. Sharifi, H. H. Shomee, S. Medya, and D. Chattopadhyay. 2025. How Older Adults Communicate their Technology Problems: Challenges and Design Opportunities. In Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '25). ACM, New York, NY, USA, Article 320, 13 pages. https://doi.org/10.1145/3706599.3719892

[2] V. S. Navale, A. A. Nadig, and N. R. Jois. 2024. Older Adults' Preference for Video-Based Tech Support: Access, Usability, and Challenges. CS 535 Final Report, University of Illinois Chicago.

[3] S. J. Czaja, N. Charness, A. D. Fisk, et al. 2006. Factors predicting the use of technology: Findings from the Center for Research and Education on Aging and Technology Enhancement (CREATE). Psychology and Aging 21, 2 (2006), 333–352. https://doi.org/10.1037/0882-7974.21.2.333

[4] Z. Wu, Z. Wu, F. Xu, et al. 2024. OS-ATLAS: A Foundation Action Model for Generalist GUI Agents. arXiv preprint arXiv:2410.23218 (2024).

[5] A. L. Xu, T. Ma, T. Liu, C. Liu, and A. Cassinelli. 2024. StoryNavi: On-Demand Narrative-Driven Reconstruction of Video Play With Generative AI. arXiv preprint (2024).

[6] Z. Li, K. You, H. Zhang, et al. 2024. Ferret-UI 2: Mastering Universal User Interface Understanding Across Platforms. arXiv preprint (2024).

[7] B. Gou, R. Wang, B. Zheng, et al. 2024. Navigating the Digital World as Humans Do: Universal Visual Grounding for GUI Agents. arXiv preprint (2024).

[8] Y. Chai, S. Huang, Y. Niu, et al. 2024. AMEX: Android Multi-annotation Expo Dataset for Mobile GUI Agents. arXiv preprint (2024). https://doi.org/10.18653/v1/2025.findings-acl.110

[9] H. Zhang, M. Gao, Z. Gan, et al. 2024. MM1.5: Methods, Analysis & Insights from Multimodal LLM Fine-tuning. arXiv preprint (2024).

[10] V. S. Navale. 2025. CS597-Automated-Visual-Tech-Support (GitHub repository). https://github.com/VibhaNavale/CS597-Automated-Visual-Tech-Support