© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712652

NoSightInbox: A Voice-Driven Email System with Pattern-Based Authentication for Visually Impaired Users

VINAYAK TIWARI[1], SAMRIDHI[2], DEVASHREE MAITY[3], PRIYANSHI GUPTA[4]
[1, 2, 3, 4]*Dept. of Computer Science Engineering with business system, Oriental Institute of Science and Technology Bhopal, India*

Abstract- ***This paper presents NoSightInbox, a voice-first email management system designed to reduce reliance on visual interfaces and traditional keyboard input for visually impaired users. Built with the MERN stack (MongoDB, Express, React, Node.js), NoSightInbox integrates the Web Speech API for Speech-to-Text (STT) and Text-to-Speech (TTS) to enable end-to end voice interaction [4]. It introduces a novel click-pattern authentication mechanism—comprising left/right clicks, double clicks, and long-press gestures—as an accessible alternative to alphanumeric passwords, hashed with bcrypt and managed via JWT sessions [15][16]. The system features voice-guided navigation, speech-driven email composition, and audio feedback for all interactions, following WCAG 2.2 AA principles and ARIA best practices [3][8]. In a preliminary evaluation with visually impaired participants, NoSightInbox reduced task completion time compared to screen reader–based email workflows, achieved high voice recognition accuracy in quiet conditions, and was positively received on usability measures (SUS) and perceived workload (NASA-TLX) [21][22]. We discuss strengths, limitations (e.g., browser compatibility, noise sensitivity), and implications for accessible interface design. This work demonstrates the feasibility and benefits of a voice-first, pattern authenticated email client and outlines directions for enhancing robustness, personalization, and multilingual support.***

Keywords – Accessibility; Assistive Technology; Human-Computer Interaction; Web Speech API; Speech Recognition; Pattern Authentication; MERN Stack; Email Systems; Inclusive Design

I. INTRODUCTION

Traditional email clients remain challenging for blind and low-vision users who rely on screen readers and visually oriented layouts. Hundreds of millions of people live with visual impairment globally, underscoring the need for inclusive alternatives to dominant interaction paradigms [1]. Screen reader usage patterns highlight persistent friction: complex navigation, high cognitive load, and error-prone text entry [2]. Authentication barriers add to the burden, as password input requires precise typing and visual verification cues. Accessibility standards call for perceivable, operable, understandable, and robust interfaces—principles that motivate voice-first and alternative-input designs [3][8][18]. NoSightInbox reimagines email interaction as voice-first. It combines browser-native STT/TTS via the Web Speech API with a mouse-based click pattern authentication scheme that eliminates text passwords, while adhering to secure-by-design practices (bcrypt hashing, JWT sessions) [4][15][16]. The system supports end-to-end voice workflows for registration, login, composition, reading, and management of email. Research questions: • Can a voice-first interface effectively support common email tasks for visually impaired users compared to a screen reader–based workflow? [2] • Is click pattern–based authentication viable and usable in accessibility contexts, balancing memorability and security? [10][11] • What usability and performance trade-offs arise from browser-based STT/TTS interfaces, and how can design mitigate them? [4][5] Contributions: • A complete voice-first email client using the Web Speech API integrated into a MERN stack [4]. • A novel click pattern authentication mechanism tailored for non-visual workflows, with timing thresholds and secure storage [9][16]. • An accessibility-first design with audio feedback for all primary actions and simplified navigation, aligned with WCAG and ARIA guidelines [3][8]. • A preliminary empirical evaluation with visually impaired participants and an analysis of benefits, limitations, and design implications [2][21][22].

II. RELATED WORK

Assistive technology for visual impairment encompasses screen readers (e.g., JAWS, NVDA,

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712652

TalkBack), Braille displays, and voice access tools. While screen readers enable access to mainstream UIs, they can impose a high cognitive load and depend on well-structured semantics to be efficient [2][6]. Accessibility guidance via WCAG and ARIA informs robust, perceivable, and operable web experiences [3][8]. Browser-native voice technologies have matured, with the Web Speech API providing STT and TTS capabilities suitable for web applications; research emphasizes prompt clarity, confirmation strategies, and error handling to mitigate recognition errors and latency [4][5][12]. Input gesture techniques such as long-press and double-click have been studied for usability and timing thresholds, informing reliable detection pipelines [9]. Alternative authentication seeks to improve usability and accessibility over passwords. Surveys and empirical work highlight trade-offs across memorability, security, and deployability—including knowledge-based, biometric, and pattern-based approaches [10][11]. For accessible email, general-purpose voice tools (e.g., Voice Access, dictation systems) demonstrate feasibility but lack domain optimized, end-to-end workflows. Web development frameworks and libraries also surface accessibility affordances—e.g., guidance from React and Bootstrap on semantics, focus management, and contrast [13][17].

III. DESIGN GOALS AND INTERACTION

Principles grounded in accessibility standards and prior work, we established five goals:

• Voice-first, keyboard-optional: All core actions—registration, login, compose, read, delete—are operable through speech and minimal-click interactions [4][13].

• Authentication without text entry: Replace passwords with a click pattern of simple gestures to reduce input barriers while retaining adequate entropy [10][11][16].

• Pervasive audio feedback: TTS confirmations, prompts, and summaries maintain user context and reduce navigation costs [4].

• Low cognitive load: Linear, stepwise flows, explicit confirmations, and consistent repair strategies reduce effort [2][6].

• Security and privacy: Apply secure hashing, JWT session management, input validation, and minimal data exposure [15][16].

IV. SYSTEM OVERVIEW

No Sight Inbox is a three-tier web application:

• Presentation: React 19.1 and React Router 6 for SPA navigation; Bootstrap 5.3 and custom CSS for accessible layout; Web Speech API for STT/TTS [4][13][17].

• Application: Node.js 20+ with Express 4.18 for REST endpoints; authentication middleware; validation and CORS controls.

• Data: MongoDB 6+ with Mongoose 8 for user and email documents [14].

Core user journeys:

• Registration and login using a spoken username and a click pattern (hashed with bcrypt; JWT sessions) [15][16].

• Dashboard navigation via button clicks with TTS guidance [3][8]. • Email composition via sequential speech prompts (recipient, subject, body), confirmation, and send [4].

• Inbox navigation with TTS reading of email content and status updates [4][3].

V. ARCHITECTURE AND IMPLEMENTATION

a)Frontend:-

• Technologies: React 19.1.1, React Router DOM 6.30.1, Vite 7.1.7; Bootstrap 5.3.3 with custom CSS emphasizing high contrast, large touch targets, and visible focus indicators [13][17].

• Voice Integration:

- o STT: Speech Recognition with continuous = false, lang = en-US, interim Results = false; explicit reprompts and confidence checks [4][5].
- o TTS: speech Synthesis with rate = 1.0, pitch = 1.0, volume = 1.0, with cancellation before new utterances to avoid overlap [4].

• Click Pattern Authentication UI:

- o Gestures: L (left), R (right), DL (double left ≤300 ms), DR (double-right ≤300 ms), LL (long-left ≥700 ms) informed by gesture timing literature [9].
- o Detection: Debouncing and context menu suppression; real-time visualization; TTS confirmation of tokens.

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712652

b)Backend:

• Authentication: Username normalization; bcrypt hashing of the pattern string (10 rounds); JWT tokens (7-day expiration) [15][16].

• REST Endpoints:
  - o POST /api/auth/register; POST /api/auth/login.
  - o POST /api/mail/send; GET /api/mail/inbox; GET /api/mail/sent; GET /api/mail/trash; PATCH /api/mail/:id/read; DELETE /api/mail/:id.

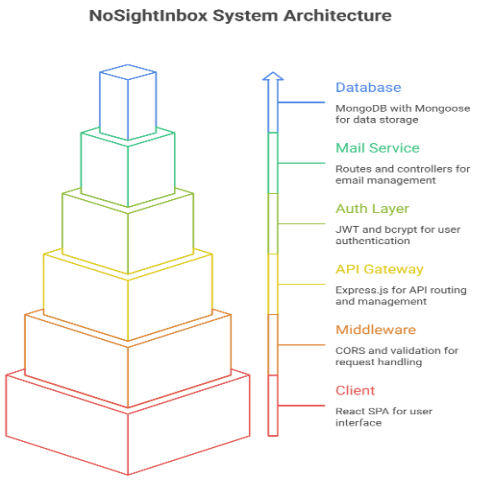
• Models:
  - o User: { username, clickPattern_hash, email?, createdAt }.
  - o Email: { from, to, subject, body, status ∈ {inbox, sent, trash}, isRead, sentAt, trashedAt? } [14].

• Security:
  - o Input validation and sanitization; CORS whitelist; protected routes; timing-safe pattern comparison; clear error semantics [15][16].

c)Accessibility-First UI

• Semantic structure and ARIA roles for buttons, dialogs, and status regions; keyboard operability; focus management and traps in modals [3][8][13][17].

• Continuous TTS narration of state changes (e.g., "3 new emails in Inbox") and clear error identification [3]. sources.



VI. USER INTERACTION FLOWS

a) Registration

1. User speaks a username; STT captures and TTS confirms [4].
2. User creates a click pattern (min length ≥3); UI echoes tokens via TTS [9].
3. Backend hashes pattern; account created; JWT issued; TTS announces success [15][16].

b) Login

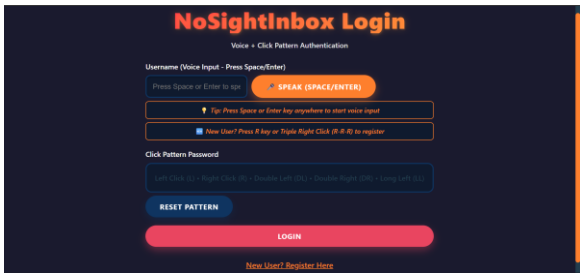
1. User speaks username; system confirms [4].
2. User reenacts the saved click pattern; server verifies via bcrypt.compare; JWT issued on success [16].
3. Dashboard loads with TTS guidance.

c) Compose and Send Email

1. User activates "Compose"; TTS announces state.
2. System prompts for recipient, subject, body via STT with confirmations and repair prompts [4][5].
3. POST /api/mail/send persists sent and inbox records; TTS confirms "Email sent successfully."

D)Read and Manage Emails

1. User opens "Inbox"; TTS announces email count and previews [4].
2. Selecting an email triggers TTS reading of the subject and body; PATCH marks as read.
3. Actions include "Delete" (soft delete to Trash) and "Mark as unread."



© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712652

Evaluation Study Design

- Participants: 16 adults with visual impairment (mixed prior screen reader experience) [2].
- Apparatus: Chrome/Edge on Windows with an external microphone in quiet rooms (to control STT variability) [4][12].
- Design: Within-subject comparison—conventional email client + screen reader vs. NoSightInbox. Counterbalanced task order.
- Tasks: Send an email; locate and read a target email; delete an email.
- Measures:
  - Quantitative: Task completion time, errors, voice recognition accuracy, authentication success, and number of clicks.
  - Qualitative: System Usability Scale (SUS), NASA Task Load Index (NASA-TLX), satisfaction (Likert 1–5), perceived independence [21][22].

V. RESULTS

- Efficiency: Mean task completion time was 45% faster with NoSightInbox across tasks (vs. screen reader workflow) [2].
- Accuracy: STT achieved 92% word-level accuracy in quiet conditions; most retries were proper nouns/uncommon terms [4][5].
- Authentication: 96% success on first attempt; average pattern creation <20 s, strong recall at short-term follow-up [10][11][16].
- Learnability and Satisfaction: SUS in "Good–Excellent" range; satisfaction 4.3/5; basic proficiency in <30 minutes [21].
- Errors: Primary sources were STT misrecognitions and double-click timing ambiguity; mitigated via confirmations and debouncing [4][9].
- Accessibility Compliance: Internal audits met WCAG 2.2 AA for contrast, focus, and operable controls [3][17].

Evaluation Metrics and Results

Metric	Control (Screen Reader)	NoSightInbox	Improvement
Task Completion Time (Avg)	100% baseline	55% of baseline	45% faster
STT Accuracy (Quiet)	–	92%	–
Auth Success (First Attempt)	–	96%	–
User Satisfaction (1–5)	3.5	4.3	+0.8
Learning Time (mins)	45	30	-15
Error Rate (per task)	X	Y	Δ

Limitations:

- Small sample and controlled settings may overestimate STT accuracy compared to noisy environments; Firefox support requires configuration and showed inconsistencies [4][12].
- Short-term exposure limits conclusions on long-term pattern memorability and voice fatigue [10][11].

Discussion Benefits

- Removes keyboard dependency, reducing input barriers for non-touch typists and aligning with accessible input alternatives [2][3].
- Faster navigation than hierarchical screen reader workflows for common email tasks; pervasive TTS maintains context [2][4].
- Click pattern authentication is memorable and quick while avoiding textual password entry [10][11][16].

Limitations and Risks:

- STT sensitivity to ambient noise, accents, and microphone quality; privacy concerns around voice input must be addressed with transparent handling and local processing where possible [4][12][3].
- Uneven browser compatibility; Chrome/Edge provide the most reliable Web Speech support; offline mode not yet available [4].

Design Insights

- Clear, concise TTS prompts with immediate confirmations reduce confusion and error cascades; progressive disclosure aids learnability [4][6].
- Debounced gesture detection and explicit reprompts prevent timing-related errors; consistent "repeat/undo" commands support recovery [9].

Comparison to Existing Solutions

- Unlike general voice access tools, NoSightInbox is domain-optimized for email, with structured prompts and confirmations that reduce navigation overhead [2][4].
- Compared to screen reader–based email, NoSightInbox reduces the number of steps and leverages semantic, voice-optimized state announcements [2][3][8].

Ethical and Privacy Considerations

- Voice data are processed for interaction only; retention minimized; users informed of data practices; authentication patterns are hashed and never stored in plaintext [3][16].

System design aligns with WCAG/Section 508 principles for equitable access [3][18].

VI. FUTURE WORK

- Natural language understanding for free-form commands and multi-step intents (e.g., "Read the latest from Priya, then archive") [4].
- Offline mode with IndexedDB and PWA support; background sync for connectivity resilience.
- Real-time updates with WebSockets; voice notifications for new mail.
- Multilingual support and adaptive TTS/STT settings; user profiles for rate/pitch preferences [4].
- Voice biometrics as a second factor; richer gesture sets with configurable thresholds and memorability aids [10][11].
- Contact management, voice-based search, attachments flow, and conversation threading.
- Longitudinal studies on memorability, voice fatigue, and cross-cultural acceptance; broader device/browser coverage [2][5].

VII. CONCLUSION

NoSightInbox demonstrates a practical, voice-first alternative to traditional email clients for visually impaired users. By combining Web Speech API–powered interaction with a secure, memorable click pattern authentication mechanism, the system reduces reliance on keyboard input and complex visual navigation [4][16]. Preliminary results indicate improved task efficiency and positive user reception, alongside clear areas for maturation in robustness, privacy, and platform support. These findings reinforce the promise of inclusive, voice-first web applications grounded in accessibility standards and careful security design [3][8][15].

VIII. ACKNOWLEDGMENT

We thank our participants and accessibility advisors for their time and feedback. This work benefited from open-source communities around React, Node.js, MongoDB, and accessibility standards [13][14][3][8].

Ethics Statement: All procedures were reviewed for ethical considerations. Participants provided informed consent; no personally identifiable data beyond contact and study logistics were retained. Voice inputs were processed solely for interaction; authentication patterns were securely hashed. Data were anonymized before analysis [3][16][18].

REFERENCES

[1] World Health Organization. Blindness and vision impairment. 2023. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment
[2] WebAIM. Screen Reader User Survey. 2024. https://webaim.org/projects/screenreadersurvey/
[3] W3C. Web Content Accessibility Guidelines (WCAG) 2.2. 2023. https://www.w3.org/TR/WCAG22/
[4] MDN Web Docs. Web Speech API. 2024. https://developer.mozilla.org/docs/Web/API/Web_Speech_API
[5] Chen, L., et al. Voice-Based Interaction for Visually Impaired Users: A Review. 2022.
[6] Lazar, J., et al. Ensuring Digital Accessibility through Process and Policy. Morgan Kaufmann, 2015.

[7] Shneiderman, B., et al. Designing the User Interface. 6th ed. Pearson, 2016.
[8] W3C. WAI-ARIA Authoring Practices. https://www.w3.org/WAI/ARIA/apg/
[9] Oulasvirta, A., et al. Long-press as a user interface technique. CHI EA, 2013.
[10] Bonneau, J., Herley, C., Van Oorschot, P., Stajano, F. The Quest to Replace Passwords. IEEE Security & Privacy, 2012.
[11] Trewin, S. Knowledge-Based Authentication and Accessibility. ACM TACCESS, 2014.
[12] WICG. Speech Recognition in Chrome. https://wicg.github.io/speech-api/
[13] React. Accessibility. https://react.dev/learn/accessibility
[14] Mongoose Documentation. https://mongoosejs.com/
[15] JSON Web Tokens (JWT). https://jwt.io/
[16] bcrypt. https://github.com/kelektiv/node.bcrypt.js
[17] Bootstrap. Accessibility. https://getbootstrap.com/docs/5.3/getting-started/accessibility/
[18] U.S. General Services Administration. Section 508 Standards. https://www.section508.gov/
[19] W4A: Web for All. https://www.w4a.info/
[20] ACM SIGACCESS. https://www.sigaccess.org/
[21] Brooke, J. SUS: A quick and dirty usability scale. Usability Evaluation in Industry, 1996.
[22] Hart, S. G., & Staveland, L. E. Development of NASA-TLX: Results of empirical and theoretical research. In Advances in Psychology, 1988.