Voice Dictation App: YC Demo Build Reference

Project Goal: Build a Wispr-like hotkey dictation app for Windows + macOS that inserts final text into the active text field (no partial transcript UI), targeting ~1000–1500ms "release hotkey → pasted text" excluding cold start.

---

1. Context Summary

Primary Focus

- **English** → **English** dictation quality + latency
- Wispr-like UX (final paste only, no streaming UI)
- Target latency: 1000–1500ms (warm endpoints, good network)

Second Priority (Later)

- Indian languages / translation
- Keep investor-ready plan, but don't build now

Key Decisions

- **Insertion method**: Clipboard + paste simulation for maximum cross-app compatibility
- **Fallback**: "Paste last transcript" hotkey when auto-insert fails
- **Architecture**: Streaming behind the scenes (client streams audio chunks to server), but UI only shows recording indicator + final inserted text
- **Server**: Serverless GPU (Cerebrium/Modal/Replicate) with warm strategy to avoid cold-start during demos
- **Models**: ASR = Whisper large-v3-turbo; Formatting = Llama 3.1 8B quantized with non-stream completion

---

2. Product Specification

User Experience Flow

1. Hold hotkey → start recording audio
2. Release hotkey → show brief "processing..." indicator (optional)
3. Text automatically inserted into currently focused input field
4. If insert fails → show toast notification with "Press <Paste Last Transcript Hotkey>"

MVP Features (In Scope)

- Windows + macOS desktop app
- Global hotkey (push-to-talk)
- Microphone selection (default mic acceptable for MVP)
- Voice commands (minimal): "new line", "new paragraph"
- Custom dictionary (local): user-defined names/terms with simple replace rules
- Insert behavior:
  - Auto insert via clipboard + paste simulation
  - Fallback: "Paste last transcript" hotkey and tray/menu button

Out of Scope (Skip for Now)

- On-device ASR/LLM
- Indic languages ASR / translation
- Fine-tuning (ASR or LLM)
- Meeting recording / file transcription
- Fancy UI (live transcript, waveform visualization)
- Team features / history sync / user accounts

---

3. Technology Stack

Desktop App Framework

Decision: Electron (TypeScript)

Rationale: Fastest shipping + best ecosystem for OS integration (global hotkeys, mic capture, clipboard manipulation, auto-updates). Footprint optimizations can be addressed later if needed.

Insertion Method

Decision: Clipboard + paste simulation (Cmd+V / Ctrl+V)

Implementation approach:

1. Save current clipboard content
2. Write transcript to clipboard
3. Simulate paste keystroke into active application
4. Restore original clipboard content

Rationale: Works across the widest set of applications and text editors. Direct accessibility-based insertion is inconsistent across apps and often requires paste fallback anyway.

Backend Deployment

Decision: Serverless GPU (Cerebrium recommended)

Configuration requirements:

- Keep ASR + LLM endpoints warm during demo windows
- Use health pings + "don't scale to zero" provider settings

- Modal equivalent knobs: min_containers, scaledown_window, buffer_containers

ASR Model

Decision: Whisper large-v3-turbo

Specifications:

- Designed for speed: 8× faster than Whisper large
- Memory efficient: ~6GB VRAM requirement
- Maintains high transcription quality for English

Formatting Model

Decision: Llama 3.1 8B (quantized to INT4/INT8)

Formatting responsibilities:

- Add punctuation and proper casing
- Apply optional style adjustments (Slack/email tone)
- Remove filler words lightly without changing meaning

LLM Streaming Decision

Decision: No token streaming required

Rationale: We only return final formatted text to client. Streaming tokens primarily benefits UI showing partial output, which we are not implementing.

Latency Target

Assumption: 1000–1500ms "release → paste" (excluding cold start)

Strategy:

- Keep endpoints warm during demo periods
- Accept 2–4s cold start initially, but avoid it during investor demos

---

4. System Architecture

Client Architecture (Electron)

Client Flow:

1. Global hotkey down → start recording
2. Stream mic audio chunks to backend over WebSocket
3. Hotkey up → send "end-of-utterance" signal
4. Receive final formatted text from server
5. Execute insertion sequence:
    - Save current clipboard
    - Write final text to clipboard
    - Simulate paste into active app
    - Restore original clipboard
6. If paste fails → show toast notification + enable "Paste last transcript"

Backend Architecture (Serverless GPU)

Two primary services (separate endpoints for easier debugging):

Service A: ASR Processing

- Receives streaming audio chunks via WebSocket
- Runs streaming-capable ASR internally
- Returns final transcript on end-of-utterance detection

Service B: Text Formatting

- Receives raw transcript from ASR service
- Runs formatter LLM (non-streaming mode)
- Returns final formatted text

Note: Services A and B can be fused into single endpoint later to reduce network overhead, but keeping separate initially aids debugging.

---

5. Paste Behavior and Fallback Strategy

Auto-Insert Mechanism

Works when:

- User's cursor focus is in normal editable text field
- Required OS permissions granted (Accessibility/Input Monitoring)

Fallback: Paste Last Transcript

Implementation pattern (inspired by Wispr Flow troubleshooting documentation):

Dedicated hotkey for manual paste:

- macOS reference: Ctrl + Cmd + V
- Windows reference: Alt + Shift + Z

Note: We can choose different hotkeys to avoid conflicts, but maintain this concept of manual fallback paste.

Expected Failure Scenarios

Auto-insert may fail due to:

- Missing OS permissions
- Hotkey conflicts with other applications
- App-level restrictions (some applications block external text insertion)
- No active cursor focus in text field

---

6. Warm Strategy for Serverless

Goal: Avoid Cold Start During Demos

Implementation strategy:

- Maintain 1 warm container for ASR endpoint
- Maintain 1 warm container for LLM endpoint
- Use periodic health ping (every 3–5 minutes) if provider scales to zero aggressively

Cost vs Latency Tradeoff

Risk assessment: Keeping endpoints warm increases billed resources compared to pure scale-to-zero approach. However, demo reliability justifies this cost during critical presentation periods.

Modal documentation explicitly notes that warm strategies trade latency for increased resource billing.

---

7. Success Metrics

Critical Latency Metrics

Metric	Description
L_total	Hotkey release → text visible in target app
L_network	Network round-trip time
L_asr	ASR finalization time
L_llm	LLM formatting time
L_paste	Local paste execution time

Table 1: Latency breakdown metrics to track

Track p50 and p90 percentiles during demo sessions.

Accuracy Metrics

For MVP demo phase:

- Subjective evaluation: "does it feel right?"
- Quick spot checks on common phrases
- Later: WER/CER on small internal evaluation set

Reliability Metrics

- Paste success rate (percentage of successful auto-inserts)
- Fallback usage rate (percentage of times "Paste last transcript" invoked)

---

8. Implementation Plan

Phase 1: Local Client + Mock Backend (1–2 days)

- Create Electron app skeleton (tray icon, settings interface)
- Implement global hotkey registration (start/stop recording)
- Build audio capture + chunking system (20–100ms frames)
- Add "Paste text now" test button to validate insertion behavior across different applications

Phase 2: Real ASR Backend (2–4 days)

- Deploy serverless ASR endpoint (Whisper large-v3-turbo)
- Implement WebSocket audio ingestion + end-of-utterance detection
- Return transcript to client application
- Add comprehensive latency instrumentation

Phase 3: Formatting Backend (1–3 days)

- Deploy serverless Llama 3.1 8B quantized endpoint
- Design and test formatting prompt with deterministic settings (low temperature)
- Integrate local dictionary replacements (pre/post formatter)

Phase 4: Demo Hardening (2–5 days)

- Implement warm strategy (keep-alive + pre-demo warmup button)
- Add "Paste last transcript" hotkey functionality
- Build clipboard save/restore mechanism
- Create permission onboarding flow:
  - macOS: Accessibility and Input Monitoring permissions
  - Windows: Administrator privilege notes

Total estimated time: 6–14 days for working MVP

---

9. Future Roadmap (Investor Story)

Indian Language Support (Phase 2)

Strategy for later implementation:

- Switch/augment ASR with AI4Bharat models (IndicConformer)
- Fine-tune using open corpora:
  - IndicVoices dataset (12,000 hours, 22 languages, CC BY 4.0 license)
  - Product-generated corrections from user feedback

Translation Capability (Phase 2)

Implementation approach:

- Add Indic → English translation after ASR stage
- Use dedicated machine translation model (IndicTrans2)
- Support 22 scheduled Indian languages

Important: Do not implement now. Keep as roadmap item for investor discussions.

---

10. Open Questions

Technical decisions requiring further investigation:

1. Which exact serverless provider to use (Cerebrium vs Modal vs Replicate)?
    - Evaluate warm container control mechanisms
    - Check GPU availability in India-adjacent regions
2. What hotkey defaults minimize OS conflicts (especially Windows)?
3. How to reliably detect "paste failed" state across different applications?
4. Optimal audio chunk size for streaming (20ms vs 50ms vs 100ms)?
5. Clipboard restore timing to avoid race conditions

---

11. Competitive Landscape Summary

Key Competitors

Product	Platform	Pricing	Key Feature
Wispr Flow	Mac/Windows	~$10-15/mo	Fastest, most polished
Superwhisper	Mac/iOS	$8.49/mo	Offline, power-user modes
VoiceInk	Mac/iOS	Open source	Privacy-first, open code
Willow Voice	Mac/iPhone	$12/mo	Enterprise features
Voice Type	Mac only	$19.99 one-time	Advanced preprocessing

Table 2: Competitive landscape overview

Market Gaps

Our key differentiators:

1. Cross-platform from day 1 (Windows + Mac, iOS/Android roadmap)
2. India-first multilingual capability (future)
3. Developer workflow integration (future)
4. Aggressive pricing: Free tier + $5/mo pro vs competitors' $8-12/mo

---

12. Risk Assessment

Technical Risks

- Cold start latency may exceed targets on some serverless platforms
- Clipboard-based paste may fail in security-sensitive applications
- Audio streaming quality degradation on poor network connections
- Permission requirements may create onboarding friction

Mitigation Strategies

- Implement aggressive warm-pool strategy during demo periods
- Build robust fallback with "Paste last transcript" hotkey
- Add audio quality indicators and retry mechanisms
- Create smooth permission request flow with clear explanations

---

13. Success Criteria for YC Demo

Must Achieve

1. Consistent 1.0–1.5s latency in demo environment
2. 95%+ paste success rate across test applications (Slack, Gmail, VS Code, browser inputs)
3. Zero crashes during 10-minute continuous demo
4. Clear, formatted output matching or exceeding native dictation quality

Nice to Have

- Custom dictionary successfully handling 5+ technical terms
- Graceful handling of background noise
- Sub-1-second latency on 50% of requests

---

14. Post-Demo Iteration Plan

Immediate (Week 1–2)

- Collect latency data from real usage
- Identify and fix top 3 paste failure scenarios
- Optimize endpoint warmup strategy based on usage patterns

Near-term (Month 1)

- Add voice commands ("new line", "period", "comma")
- Implement searchable history of past transcriptions
- Build basic analytics dashboard (latency, accuracy, usage)

Medium-term (Month 2–3)

- Begin Indian language support research
- Explore developer-specific features (code formatting, technical vocabulary)
- Add app-specific formatting profiles

References

1. OpenAI. (2024). Whisper large-v3-turbo model. Retrieved from https://huggingface.co/openai/whisper-large-v3-turbo
2. Modal Labs. (2026). Cold start performance documentation. Retrieved from https://modal.com/docs/guide/cold-start
3. Wispr Flow. (2026). Text not pasting or inserting - troubleshooting guide. Retrieved from https://docs.wisprflow.ai/articles/4481008749-text-not-pasting-or-inserting
4. AI4Bharat. (2024). IndicVoices dataset. Retrieved from https://huggingface.co/datasets/ai4bharat/IndicVoices
5. Wispr Flow. (2025). Technical challenges behind Flow. Retrieved from https://wisprflow.ai/post/technical-challenges

Voice Dictation App: YC Demo Build Reference

Project Goal: Build a Wispr-like hotkey dictation app for Windows + macOS that inserts final text into the active text field (no partial transcript UI), targeting ~1000–1500ms "release hotkey → pasted text" excluding cold start.

---

1. Context Summary

Primary Focus

- **English** → **English** dictation quality + latency
- Wispr-like UX (final paste only, no streaming UI)
- Target latency: 1000–1500ms (warm endpoints, good network)

Second Priority (Later)

- Indian languages / translation
- Keep investor-ready plan, but don't build now

Key Decisions

- **Insertion method**: Clipboard + paste simulation for maximum cross-app compatibility
- **Fallback**: "Paste last transcript" hotkey when auto-insert fails
- **Architecture**: Streaming behind the scenes (client streams audio chunks to server), but UI only shows recording indicator + final inserted text
- **Server**: Serverless GPU (Cerebrium/Modal/Replicate) with warm strategy to avoid cold-start during demos

- **Models**: ASR = Whisper large-v3-turbo; Formatting = Llama 3.1 8B quantized with non-stream completion

---

2. Product Specification

User Experience Flow

1. Hold hotkey → start recording audio
2. Release hotkey → show brief "processing..." indicator (optional)
3. Text automatically inserted into currently focused input field
4. If insert fails → show toast notification with "Press <Paste Last Transcript Hotkey>"

MVP Features (In Scope)

- Windows + macOS desktop app
- Global hotkey (push-to-talk)
- Microphone selection (default mic acceptable for MVP)
- Voice commands (minimal): "new line", "new paragraph"
- Custom dictionary (local): user-defined names/terms with simple replace rules
- Insert behavior:
  - Auto insert via clipboard + paste simulation
  - Fallback: "Paste last transcript" hotkey and tray/menu button

Out of Scope (Skip for Now)

- On-device ASR/LLM
- Indic languages ASR / translation
- Fine-tuning (ASR or LLM)
- Meeting recording / file transcription
- Fancy UI (live transcript, waveform visualization)
- Team features / history sync / user accounts

---

3. Technology Stack

Desktop App Framework

Decision: Electron (TypeScript)

Rationale: Fastest shipping + best ecosystem for OS integration (global hotkeys, mic capture, clipboard manipulation, auto-updates). Footprint optimizations can be addressed later if needed.

Insertion Method

Decision: Clipboard + paste simulation (Cmd+V / Ctrl+V)

Implementation approach:

1. Save current clipboard content
2. Write transcript to clipboard
3. Simulate paste keystroke into active application

4. Restore original clipboard content

Rationale: Works across the widest set of applications and text editors. Direct accessibility-based insertion is inconsistent across apps and often requires paste fallback anyway.

Backend Deployment

Decision: Serverless GPU (Cerebrium recommended)

Configuration requirements:

- Keep ASR + LLM endpoints warm during demo windows
- Use health pings + "don't scale to zero" provider settings
- Modal equivalent knobs: min_containers, scaledown_window, buffer_containers

ASR Model

Decision: Whisper large-v3-turbo

Specifications:

- Designed for speed: 8× faster than Whisper large
- Memory efficient: ~6GB VRAM requirement
- Maintains high transcription quality for English

Formatting Model

Decision: Llama 3.1 8B (quantized to INT4/INT8)

Formatting responsibilities:

- Add punctuation and proper casing
- Apply optional style adjustments (Slack/email tone)
- Remove filler words lightly without changing meaning

LLM Streaming Decision

Decision: No token streaming required

Rationale: We only return final formatted text to client. Streaming tokens primarily benefits UI showing partial output, which we are not implementing.

Latency Target

Assumption: 1000–1500ms "release → paste" (excluding cold start)

Strategy:

- Keep endpoints warm during demo periods
- Accept 2–4s cold start initially, but avoid it during investor demos

---

4. System Architecture

Client Architecture (Electron)

Client Flow:

1. Global hotkey down → start recording
2. Stream mic audio chunks to backend over WebSocket
3. Hotkey up → send "end-of-utterance" signal
4. Receive final formatted text from server
5. Execute insertion sequence:
    - Save current clipboard
    - Write final text to clipboard
    - Simulate paste into active app
    - Restore original clipboard
6. If paste fails → show toast notification + enable "Paste last transcript"

Backend Architecture (Serverless GPU)

Two primary services (separate endpoints for easier debugging):

Service A: ASR Processing

- Receives streaming audio chunks via WebSocket
- Runs streaming-capable ASR internally
- Returns final transcript on end-of-utterance detection

Service B: Text Formatting

- Receives raw transcript from ASR service
- Runs formatter LLM (non-streaming mode)
- Returns final formatted text

Note: Services A and B can be fused into single endpoint later to reduce network overhead, but keeping separate initially aids debugging.

---

5. Paste Behavior and Fallback Strategy

Auto-Insert Mechanism

Works when:

- User's cursor focus is in normal editable text field
- Required OS permissions granted (Accessibility/Input Monitoring)

Fallback: Paste Last Transcript

Implementation pattern (inspired by Wispr Flow troubleshooting documentation):

Dedicated hotkey for manual paste:

- macOS reference: Ctrl + Cmd + V
- Windows reference: Alt + Shift + Z

Note: We can choose different hotkeys to avoid conflicts, but maintain this concept of manual fallback paste.

Expected Failure Scenarios

Auto-insert may fail due to:

- Missing OS permissions
- Hotkey conflicts with other applications
- App-level restrictions (some applications block external text insertion)
- No active cursor focus in text field

---

6. Warm Strategy for Serverless

Goal: Avoid Cold Start During Demos

Implementation strategy:

- Maintain 1 warm container for ASR endpoint
- Maintain 1 warm container for LLM endpoint
- Use periodic health ping (every 3–5 minutes) if provider scales to zero aggressively

Cost vs Latency Tradeoff

Risk assessment: Keeping endpoints warm increases billed resources compared to pure scale-to-zero approach. However, demo reliability justifies this cost during critical presentation periods.

Modal documentation explicitly notes that warm strategies trade latency for increased resource billing.

---

7. Success Metrics

Critical Latency Metrics

Metric	Description
L_total	Hotkey release → text visible in target app
L_network	Network round-trip time
L_asr	ASR finalization time
L_llm	LLM formatting time
L_paste	Local paste execution time

Table 3: Latency breakdown metrics to track

Track p50 and p90 percentiles during demo sessions.

Accuracy Metrics

For MVP demo phase:

- Subjective evaluation: "does it feel right?"
- Quick spot checks on common phrases
- Later: WER/CER on small internal evaluation set

Reliability Metrics

- Paste success rate (percentage of successful auto-inserts)
- Fallback usage rate (percentage of times "Paste last transcript" invoked)

---

8. Implementation Plan

Phase 1: Local Client + Mock Backend (1–2 days)

- Create Electron app skeleton (tray icon, settings interface)
- Implement global hotkey registration (start/stop recording)
- Build audio capture + chunking system (20–100ms frames)
- Add "Paste text now" test button to validate insertion behavior across different applications

Phase 2: Real ASR Backend (2–4 days)

- Deploy serverless ASR endpoint (Whisper large-v3-turbo)
- Implement WebSocket audio ingestion + end-of-utterance detection
- Return transcript to client application
- Add comprehensive latency instrumentation

Phase 3: Formatting Backend (1–3 days)

- Deploy serverless Llama 3.1 8B quantized endpoint
- Design and test formatting prompt with deterministic settings (low temperature)
- Integrate local dictionary replacements (pre/post formatter)

Phase 4: Demo Hardening (2–5 days)

- Implement warm strategy (keep-alive + pre-demo warmup button)
- Add "Paste last transcript" hotkey functionality
- Build clipboard save/restore mechanism
- Create permission onboarding flow:
  - macOS: Accessibility and Input Monitoring permissions
  - Windows: Administrator privilege notes

Total estimated time: 6–14 days for working MVP

---

9. Future Roadmap (Investor Story)

Indian Language Support (Phase 2)

Strategy for later implementation:

- Switch/augment ASR with AI4Bharat models (IndicConformer)
- Fine-tune using open corpora:
    - IndicVoices dataset (12,000 hours, 22 languages, CC BY 4.0 license)
    - Product-generated corrections from user feedback

Translation Capability (Phase 2)

Implementation approach:

- Add Indic → English translation after ASR stage
- Use dedicated machine translation model (IndicTrans2)
- Support 22 scheduled Indian languages

Important: Do not implement now. Keep as roadmap item for investor discussions.

---

10. Open Questions

Technical decisions requiring further investigation:

1. Which exact serverless provider to use (Cerebrium vs Modal vs Replicate)?
    - Evaluate warm container control mechanisms
    - Check GPU availability in India-adjacent regions
2. What hotkey defaults minimize OS conflicts (especially Windows)?
3. How to reliably detect "paste failed" state across different applications?
4. Optimal audio chunk size for streaming (20ms vs 50ms vs 100ms)?
5. Clipboard restore timing to avoid race conditions

---

11. Competitive Landscape Summary

Key Competitors

Product	Platform	Pricing	Key Feature
Wispr Flow	Mac/Windows	~$10-15/mo	Fastest, most polished
Superwhisper	Mac/iOS	$8.49/mo	Offline, power-user modes
VoiceInk	Mac/iOS	Open source	Privacy-first, open code
Willow Voice	Mac/iPhone	$12/mo	Enterprise features
Voice Type	Mac only	$19.99 one-time	Advanced preprocessing

Table 4: Competitive landscape overview

Market Gaps

Our key differentiators:

1. Cross-platform from day 1 (Windows + Mac, iOS/Android roadmap)
2. India-first multilingual capability (future)
3. Developer workflow integration (future)
4. Aggressive pricing: Free tier + $5/mo pro vs competitors' $8-12/mo

---

12. Risk Assessment

Technical Risks

- Cold start latency may exceed targets on some serverless platforms
- Clipboard-based paste may fail in security-sensitive applications
- Audio streaming quality degradation on poor network connections
- Permission requirements may create onboarding friction

Mitigation Strategies

- Implement aggressive warm-pool strategy during demo periods
- Build robust fallback with "Paste last transcript" hotkey
- Add audio quality indicators and retry mechanisms
- Create smooth permission request flow with clear explanations

---

13. Success Criteria for YC Demo

Must Achieve

1. Consistent 1.0–1.5s latency in demo environment
2. 95%+ paste success rate across test applications (Slack, Gmail, VS Code, browser inputs)
3. Zero crashes during 10-minute continuous demo
4. Clear, formatted output matching or exceeding native dictation quality

Nice to Have

- Custom dictionary successfully handling 5+ technical terms
- Graceful handling of background noise
- Sub-1-second latency on 50% of requests

---

14. Post-Demo Iteration Plan

Immediate (Week 1–2)

- Collect latency data from real usage
- Identify and fix top 3 paste failure scenarios
- Optimize endpoint warmup strategy based on usage patterns

Near-term (Month 1)

- Add voice commands ("new line", "period", "comma")
- Implement searchable history of past transcriptions
- Build basic analytics dashboard (latency, accuracy, usage)

Medium-term (Month 2–3)

- Begin Indian language support research
- Explore developer-specific features (code formatting, technical vocabulary)
- Add app-specific formatting profiles

References

[1] OpenAI. (2024). Whisper large-v3-turbo model. Retrieved from https://huggingface.co/openai/whisper-large-v3-turbo

[2] Modal Labs. (2026). Cold start performance documentation. Retrieved from https://modal.com/docs/guide/cold-start

[3] Wispr Flow. (2026). Text not pasting or inserting - troubleshooting guide. Retrieved from https://docs.wisprflow.ai/articles/4481008749-text-not-pasting-or-inserting

[4] AI4Bharat. (2024). IndicVoices dataset. Retrieved from https://huggingface.co/datasets/ai4bharat/IndicVoices

[5] Wispr Flow. (2025). Technical challenges behind Flow. Retrieved from https://wisprflow.ai/post/technical-challenges