Lilapu Desktop App — PRD

Problem

Psycholodzy, coache i mentorzy prowadzą sesje online (Zoom, Meet, Teams) i potrzebują transkrypcji obu stron rozmowy. Obecna wersja web **nie może przechwycić głosu rozmówcy** ze słuchawek — mikrofon łapie tylko użytkownika. Desktop app rozwiązuje to przez przechwytywanie audio systemowego.

Cele produktu

1. **Nagrywanie obu stron** rozmowy online (mikrofon + system audio)
2. **Zero konfiguracji** — działa od razu po instalacji, bez dodatkowych sterowników
3. **Pełna kompatybilność** z dowolną apką video (Zoom, Meet, Teams, FaceTime)
4. **E2EE zachowane** — te same mechanizmy szyfrowania co wersja web
5. **Identyczny interfejs** — użytkownik znający wersję web czuje się jak w domu
6. **Zaawansowany OCR** — screenshot->OCR, drag&drop, macOS Vision offline

---

Strategia multi-platformowa

Platforma	Rozwiązanie	System audio	OCR	Nagrywanie mic
macOS	Desktop app (Tauri)	[TAK]	[TAK] Pełne (screenshot, Vision)	[TAK]
Windows	Desktop app (Tauri)	[TAK]	[TAK] Pełne (screenshot)	[TAK]
iOS / Android / tablet	**PWA** (już istnieje [TAK])	[NIE]	[TAK] Kamera -> RunPod	[TAK]

> [!NOTE] PWA obsługuje mobile/tablet. Desktop app jest potrzebna **wyłącznie** dla system audio capture i zaawansowanego OCR.

Architektura

```
graph TB
    subgraph "Lilapu Desktop App"
        UI["React UI (WebView)"]
        BRIDGE["Native Bridge (Tauri)"]
```

```
    MIC["Mikrofon Capture"]
    SYS["System Audio Capture"]
    MIXER["Audio Mixer"]
    OCR["OCR Engine"]
    VISION["macOS Vision / Screenshot"]
end

subgraph "Zewnętrzne usługi"
    CONVEX["Convex Backend"]
    WHISPER["RunPod Whisper"]
    BIELIK["RunPod Bielik"]
    GOTOCR["RunPod GOT-OCR"]
end

MIC --> MIXER
SYS --> MIXER
MIXER --> UI
UI --> BRIDGE
BRIDGE --> CONVEX
UI --> WHISPER
UI --> BIELIK
VISION --> OCR
OCR --> GOTOCR
OCR --> BIELIK
```

Źródła audio

Scenariusz	Mikrofon	System Audio	Wynik
Spotkanie online (słuchawki)	[TAK] Twój głos	[TAK] Głos rozmówcy	Pełna transkrypcja
Spotkanie online (głośnik)	[TAK] Oba głosy	[!] Opcjonalne	Pełna transkrypcja
Spotkanie na żywo (gabinet)	[TAK] Oba głosy	[NIE] Nie potrzebne	Pełna transkrypcja

---

Technologia

Rekomendacja: Tauri v2

Kryterium	Electron	Tauri v2
Rozmiar instalki	~150 MB	**~5 MB**
RAM usage	~300 MB	**~50 MB**
Natywne API audio	Node.js addon	**Rust (cpal/rodio)**
Auto-update	electron-updater	**wbudowany**
Podpisanie kodu	Tak	**Tak**
macOS system audio	Wymaga BlackHole	**ScreenCaptureKit (natywny)**

Kryterium	Electron	Tauri v2
Windows system audio	WASAPI loopback	**WASAPI loopback**

> [!IMPORTANT] **Tauri v2** jest rekomendowany ze względu na mały rozmiar (5 MB vs 150 MB), niski RAM, i natywne API Rust do przechwytywania audio. Frontend (React) działa bez zmian w WebViewie.

Audio Capture — mechanizmy per platforma

macOS (Sonoma 14.4+): - `ScreenCaptureKit` — natywny API Apple do przechwytywania audio z apek - Nie wymaga BlackHole ani żadnych dodatkowych sterowników - Użytkownik musi jednorazowo zatwierdzić uprawnienie “Screen & System Audio Recording”

macOS (starsze): - Fallback na BlackHole (auto-instalacja z poziomu apki lub prompt)

Windows: - WASAPI Loopback — wbudowany w Windows, zero instalacji - Przechwytuje audio wyjściowe systemu

---

User Flow

Pierwsza instalacja

```
1. Pobiera z lilapu.com/pobierz
   └─ macOS: Lilapu.dmg (5 MB)
   └─ Windows: Lilapu-Setup.exe (5 MB)

2. Instalacja
   └─ macOS: przeciąga do /Applications
   └─ Windows: Next -> Next -> Finish

3. Pierwsze uruchomienie
   └─ "Pozwolić Lilapu na mikrofon?" -> Zezwól
   └─ macOS: "Pozwolić na nagrywanie audio?" -> Zezwól
   └─ Logowanie (Clerk) -> Hasło szyfrowania (E2EE)

4. Gotowe — ikona w:
   └─ macOS: menu bar (górny pasek)
   └─ Windows: system tray
```

Codzienny use case — sesja online

```
1. Otwiera Zoom/Meet/Teams -> rozpoczyna spotkanie
2. Klika ikonę Lilapu w menu bar -> " Nagrywaj rozmowę"
3. Wybiera tryb:  Mikrofon |  Mikrofon + System Audio
4. Lilapu nagrywa w tle — miniaturowe okno z timerem
5. Po spotkaniu -> Stop -> Whisper transkrybuje -> Bielik poleruje
6. Podsumowanie sesji generuje się automatycznie
7. Może dodać do notatek -> dostępne w Chat AI
```

Codzienny use case — gabinet (na żywo)

```
1. Klika " Nagrywaj" -> tryb:  Mikrofon (jak w wersji web)
2. Mikrofon łapie obie osoby w pokoju
3. Reszta identyczna — transkrypcja, polishing, podsumowanie
```

Funkcje MVP

P0 — Muszą być w v1.0

#	Funkcja	Opis
1	**Nagrywanie z mikrofonu**	Identyczne jak wersja web
2	**Nagrywanie system audio**	Przechwytuje audio z dowolnej apki
3	**MIX mikrofon + system**	Łączy oba strumienie w jeden
4	**Menu bar / System tray**	Szybki dostęp, timer nagrywania
5	**Logowanie (Clerk)**	Ten sam system auth co web
6	**E2EE**	Ten sam moduł crypto
7	**Transkrypcja (Whisper)**	RunPod, identyczna jak web
8	**Bielik polishing + summary**	Jak w web
9	**Dashboard**	Pełny interfejs: transkrypcje, notatki, Czat AI
10	**Auto-update**	Tauri wbudowany updater
11	**OCR import zdjęcia**	Jak w web — plik -> RunPod GOT-OCR -> Bielik
12	**Screenshot -> OCR**	Cmd+Shift+S zaznacza fragment ekranu -> OCR rozpoznaje tekst
13	**Drag & drop -> OCR**	Przeciąga zdjęcie na ikonę Lilapu -> OCR

P1 — Wersja 1.1

#	Funkcja	Opis
1	**Diaryzacja (kto mówi)**	Osobne ścieżki: mikrofon = Ty, system = Rozmówca
2	**Hotkey**	Globalny skrót klawiszowy (np. Cmd+Shift+R)
3	**Floating widget**	Mały pływający timer podczas nagrywania
4	**macOS Vision (offline OCR)**	Apple Vision Framework — OCR lokalnie, bez RunPod
5	**Windows OCR (offline)**	Windows.Media.Ocr — OCR lokalnie
6	**Offline mode**	Lokalna transkrypcja Whisper (whisper.cpp) gdy brak internetu

#	Funkcja	Opis
7	**Kamera na żywo -> OCR**	Podgląd kamery, celowanie w notatkę, klik -> OCR

P2 — Przyszłość

#	Funkcja
1	Real-time live transcription (WebSocket)
2	Integracja z kalendarzem (auto-nagrywanie)
3	iOS companion app (mikrofon only)
4	Wersja Linux

Struktura projektu

```
lilapu/
├── web/                    # Obecna wersja web (Next.js + Convex)
├── desktop/                # NOWY — Tauri app
│   ├── src-tauri/          # Rust backend
│   │   ├── src/
│   │   │   ├── main.rs     # Entry point
│   │   │   ├── audio.rs    # System audio capture
│   │   │   ├── mixer.rs    # Mic + system audio mixer
│   │   │   └── tray.rs     # Menu bar / system tray
│   │   ├── Cargo.toml
│   │   └── tauri.conf.json
│   └── src/                # Frontend (React — shared z web)
│       ├── App.tsx
│       └── components/     # Reużywane komponenty z web/
├── convex/                 # Backend (bez zmian)
└── runpod-*/               # AI endpoints (bez zmian)
```

Estymacja

Faza	Czas	Opis
Setup Tauri + WebView	1 dzień	Scaffold, React w WebView, Convex połączenie
System audio capture	2 dni	ScreenCaptureKit (macOS) + WASAPI (Windows)
Audio mixer + nagrywanie	1 dzień	Łączenie mikrofon + system, WAV encoding
Menu bar / tray	0.5 dnia	Ikona, timer, start/stop
Packaging + signing	1 dzień	.dmg, .exe, code signing, auto-update

Faza	Czas	Opis
Testowanie	1.5 dnia	macOS + Windows manual QA
TOTAL MVP	**~7 dni**	

Ryzyka

Ryzyko	Prawdopodobieństwo	Mitygacja
macOS pre-Sonoma brak ScreenCaptureKit	Średnie	Fallback: BlackHole auto-install
Apple code signing wymaga $99/rok Developer ID	[TAK] Pewne	Konto Apple Developer potrzebne
Windows SmartScreen blokuje nieznane apki	Wysokie	EV code signing certificate (~$200/rok)
Tauri v2 audio cpal bugs	Niskie	Fallback: portaudio binding

Konkurencja

Produkt	System Audio	Diaryzacja	E2EE	Polski	Cena
Otter.ai	[TAK]	[TAK]	[NIE]	[NIE]	$16/mies
Fireflies.ai	[TAK]	[TAK]	[NIE]	[!] słabo	$18/mies
Krisp	[TAK]	[NIE]	[NIE]	[NIE]	$12/mies
Tactiq	[!] Extension	[NIE]	[NIE]	[NIE]	$12/mies
Lilapu	[TAK] (MVP)	[TAK] (P1)	**[TAK]**	**[TAK]**	TBD

[!TIP] **Przewaga Lilapu**: jedyne narzędzie z **E2EE + natywną obsługą polskiego** (Bielik). Dla psychologów i prawników E2EE to requirement, nie feature.