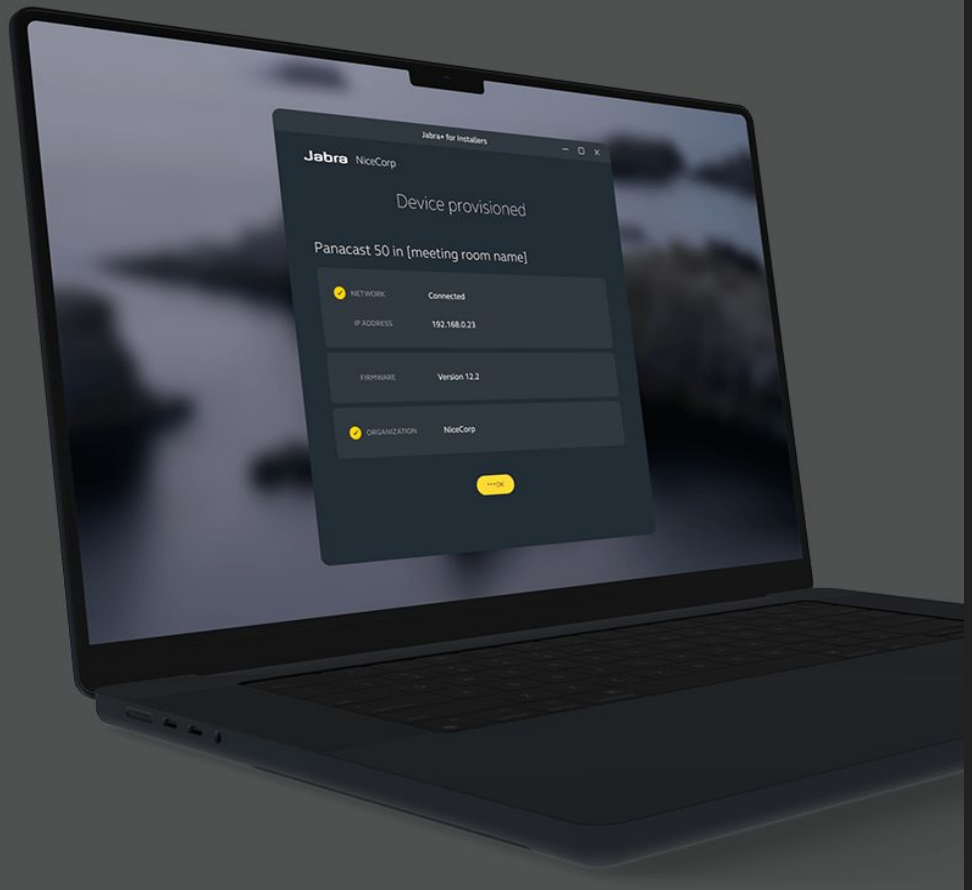
AudiSense

a case study

WPF to Blazor journey

Who am I?

Dogfooding and EAP aficionado



I was there before

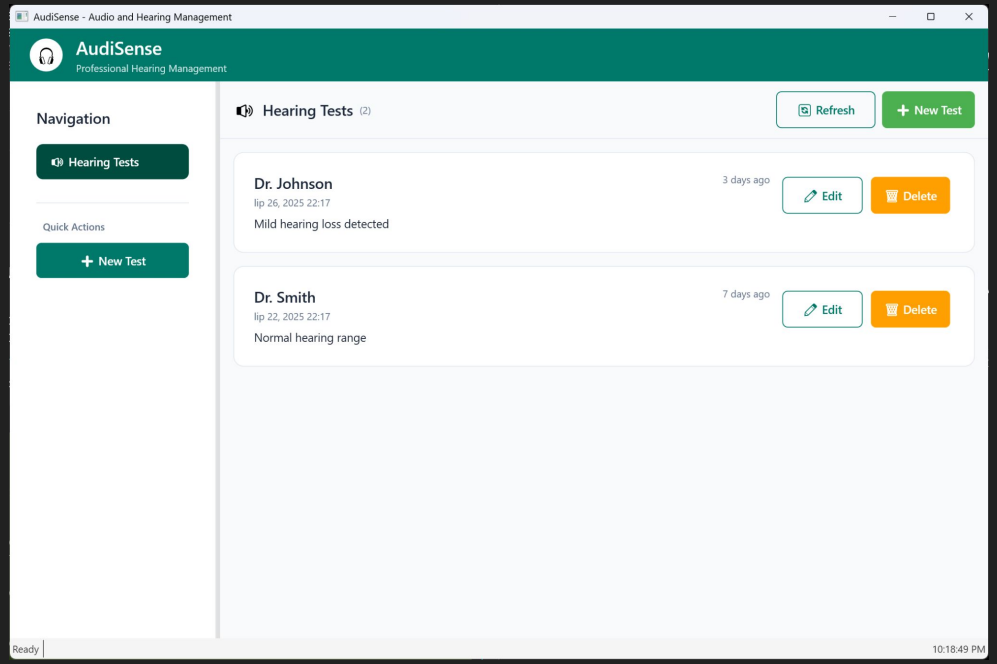
From a full-stack web developers band to native-first developers team of network geeks and meeting room power users

Why are we here? AudiSense

AudiSense is a (study case) application for hearing tests built with WPF and .NET 9.0. This application serves hearing tests to users, allowing them to conduct hearing tests which are saved to the system and can be reviewed and updated later. The solution implements a multi-client architecture with a REST API backend and WPF desktop client.

Why are we here? AudiSense

It’s there, in the the source code repository, check it out!



Why are we here? AudiSense

From Desktop to Web

Challenge: WPF app needs broader reach
Solution: Progressive migration to Blazor WebAssembly

Our Goals

- **Expand accessibility** - Web-based delivery
- **Modernize stack** - Latest .NET + web technologies
- **Grow team skills** - Web development competencies
- **Preserve investment** - Reuse existing business logic

Success Metric

🎯 **Every development cycle advances both application capabilities and team web skills**

Technical Decision Collaboration 🤝

Architecture Decision Records (ADRs)

Documenting key technical choices for transparent team collaboration

Technical Decision Collaboration 🤝

Migration Strategy Overview

Foundation (ADRs 1-6): Migration approach, architecture assessment, Blazor WebAssembly selection

Implementation (ADRs 7-10): Styling, tooling (VS2022), PWA offline support, user-focused testing

Experience (ADRs 11-14): Security model, UX migration, standalone PWA mode, performance optimization

Technical Decision Collaboration 🤝

Key Decisions Made

✅ **Progressive migration** - Preserve WPF investment while adding web capabilities

✅ **Blazor WebAssembly** - Offline-capable, client-side performance

✅ **Standalone PWA** - Native desktop-like experience

✅ **User-focused testing** - Accessibility-first, semantic testing approach

✅ **Performance optimization** - AOT compilation, lazy loading, monitoring

Technical Decision Collaboration 🤝

Collaboration Benefits

📋 **Transparent decision-making**

🧠 **Shared technical understanding**

📈 **Knowledge transfer built-in**

Implementation Details & Project Timeline 🚀

Migration Architecture Overview

Service-Preserved Component Model - Maximizing code reuse while modernizing UI

Code Reusability Assessment

🟢 **90-100% Reusable**: Domain entities, API controllers, business rules

🟡 **70-90% Adaptable**: HTTP clients, business services (remove WPF dependencies)

🔄 **Pattern-Mappable**: ViewModels → Blazor components, UserControls → `.razor` files

📊 **Overall Reuse**: 75-80% of existing codebase preserved

Implementation Details & Project Timeline 🚀

Implementation Phases

Phase 1: Foundation (Weeks 1-2)

✅ Base architecture setup with Clean Architecture principles

✅ Blazor WebAssembly + PWA configuration

✅ Navigation system (replace WPF windows with Blazor Router)

✅ Core shared services migration

Implementation Phases

Phase 2: Component Migration (Weeks 3-6)

🔄 Transform ViewModels to Blazor component state (`@code` blocks)

🔄 Convert UserControls to `.razor` components

🔄 Implement data binding with reactive updates via `StateHasChanged()`

🔄 Replace Commands with event handlers

Implementation Details & Project Timeline 🚀

Implementation Phases

Phase 3: Enhancement (Weeks 7-8)

⚡ PWA features (offline support, native-like experience)

📱 Responsive design with CSS Grid/Flexbox

🎯 Performance optimization (AOT, lazy loading, bundle size < 10MB)

Implementation Details & Project Timeline 🚀

Key Technical Transformations

WPF Pattern	Blazor Equivalent	Implementation
ViewModels	Component state	`@code` blocks with local state
UserControls	Razor components	`.razor` files with parameters
Commands	Event handlers	`@onclick="HandleClick"`
Data binding	Parameter binding	`@bind-Value="Property"`
Navigation	Router + pages	`@page` directives + NavigationManager

Code reviews **Objectives**

LET’S START WITH A STATEMENT:

Team already has strong foundational practices

Key Outcome:

Each pull request should advance both the application's web capabilities and the team's web development competencies **simultaneously**.

Code reviews **PR Templates**

The templates maintain the key objective:

"Each PR should advance both web capabilities and team competencies" while being practical for daily use by developers who already have strong .NET foundations.

Both for Azure DevOps and Github

Code reviews Values beyond standard practices

Web-Native Pattern Enforcement

- **Objective**: Ensure code embraces web paradigms rather than forcing desktop patterns into web context
- **Review Focus**: Are we using web-appropriate solutions or desktop workarounds?
- **Example**: Using CSS Grid/Flexbox for layout vs trying to recreate WPF panels

Progressive Enhancement Validation

- **Objective**: Verify the migration unlocks new capabilities, not just functional equivalence
- **Review Focus**: Does this code make the application more capable than the desktop version?
- **Example**: Adding responsive design, offline capabilities, or accessibility features

Code reviews Values beyond standard practices

Modern Web Standards Adoption

- **Objective**: Build team competency in current web development practices, not legacy approaches
- **Review Focus**: Are we leveraging modern web technologies appropriately?
- **Example**: Using semantic HTML, CSS custom properties, current JavaScript features

Performance-First Web Thinking

- **Objective**: Apply web-specific performance considerations different from desktop applications
- **Review Focus**: Does this follow web performance best practices?
- **Example**: Component rendering optimization, lazy loading, bundle size awareness

Workshops **Values**

Leverage Existing Skills

- Utilize the team's .NET/C# expertise to ensure smooth transition.
- Focus on familiar concepts while introducing web-specific practices.

Clear Progression Path

- Begin with foundational topics and gradually advance to more complex concepts.
- Ensure each workshop builds on the previous to reinforce learning.

Workshops **Values**

Hands-On Practice

- Encourage practical exercises that simulate real-world scenarios.
- Provide exercises that reflect common challenges in web development.

Fostering Innovation

- Encourage the team to think beyond desktop paradigms and embrace modern web capabilities.
- Introduce new tools and technologies to stimulate creativity and innovation.

Workshops

Workshop Formats and Delivery Options
Agile and Lean

- **Structured Series**: Follow 1-20 progression for comprehensive migration knowledge
- **Ad-hoc Sessions**: Target specific competency gaps or emerging team challenges
- **Problem-driven**: Address real project blockers with relevant workshop content

Workshops Core Workshop Topics

Foundation (Workshops 1-6)

Layout & Styling (Workshops 7-8)

Advanced Web Features (Workshops 9-12)

Production-Ready Development (Workshops 13-20)

Thank you!