NeuroBreath Technical Decisions Log

Purpose: Document key architectural, technical, and design decisions made during development. This log helps maintain consistency, onboard new developers, and provide context for future changes.

Decision Log Format

Each decision entry follows this structure:

```
### [DECISION-XXX] Title
**Date**: YYYY-MM-DD
**Status**: Accepted | Superseded | Deprecated
**Context**: Why this decision was needed
**Decision**: What was decided
**Alternatives Considered**: Other options evaluated
**Consequences**: Trade-offs and implications
**Related Decisions**: Links to other decisions
```

🏛️ Architecture Decisions

[DECISION-001] Next.js 14 App Router

Date: 2024-12-15
Status: Accepted
Context: Needed a modern React framework with SSR, file-based routing, and strong TypeScript support. Considered Next.js Pages Router, Remix, and Astro.

Decision: Use Next.js 14 with App Router for the web application.

Alternatives Considered:
1. **Next.js Pages Router**: More mature, but App Router offers better data fetching patterns
2. **Remix**: Strong data loading, but smaller ecosystem and learning curve
3. **Astro**: Excellent for static sites, but limited for dynamic features we need (auth, real-time tracking)

Consequences:
- ✅ Benefits: Server Components reduce client-side JS, built-in loading states, streaming SSR
- ❌ Trade-offs: App Router is newer (some ecosystem packages incompatible), steeper learning curve
- ⚠️ Risks: Breaking changes in future Next.js versions

Related Decisions: [DECISION-002], [DECISION-010]

[DECISION-002] Cloudflare Pages + Workers Deployment

Date: 2024-12-20
Status: Accepted

Context: Needed hosting with global CDN, edge computing for SSR, and low cost. Considered Vercel, Netlify, and self-hosted options.

Decision: Deploy on Cloudflare Pages with Workers for SSR, targeting www.neurobreath.co.uk.

Alternatives Considered:
1. **Vercel**: Native Next.js support, but higher cost at scale ($20/month + usage)
2. **Netlify**: Good DX, but weaker SSR support for Next.js
3. **AWS Amplify**: More control, but complex setup and AWS pricing unpredictability
4. **Self-Hosted (VPS)**: Full control, but requires DevOps maintenance

Consequences:
- ✅ Benefits: $5/month Workers plan, global CDN (200+ cities), 99.99% uptime SLA, built-in DDoS protection
- ❌ Trade-offs: Workers have 50ms CPU limit (requires efficient code), some Next.js features need adapters
- ⚠️ Risks: Vendor lock-in to Cloudflare ecosystem

Implementation Notes:
- Use `@cloudflare/next-on-pages` adapter for SSR compatibility
- Configure apex domain redirect: `neurobreath.co.uk` → `www.neurobreath.co.uk`
- Leverage Cloudflare R2 for media storage (future)

Related Decisions: [DECISION-001], [DECISION-003]

[DECISION-003] Monorepo Structure

Date: 2024-12-25
Status: Accepted
Context: Anticipating multi-platform expansion (web, mobile, serverless APIs). Needed a scalable project structure that supports code sharing and independent deployments.

Decision: Organize as monorepo with `/web`, `/shared`, `/serverless`, `/flutter_app` directories.

Alternatives Considered:
1. **Single Repository per Platform**: Simpler initially, but duplicates shared code (design tokens, data files)
2. **Turborepo/Nx Workspace**: More sophisticated tooling, but overkill for current team size
3. **Polyrepo with npm Packages**: Shared code as npm packages, but complex versioning and publishing

Consequences:
- ✅ Benefits: Single source of truth for shared data/assets, easier refactoring across platforms, consistent versioning
- ❌ Trade-offs: Larger repository size, requires discipline to avoid circular dependencies
- ⚠️ Risks: Build complexity increases as more platforms added

Directory Structure:

```
neurobreath-platform/
├── web/              # Next.js web app (current)
├── shared/           # Data, design tokens, assets
├── serverless/       # Cloudflare Workers/Functions
├── flutter_app/      # Mobile app (future)
└── docs/             # Documentation
```

Related Decisions: [DECISION-001], [DECISION-002]

[DECISION-004] LocalStorage for Progress Tracking (Phase 1)

Date: 2024-12-18
Status: Accepted (Temporary)
Context: Need progress tracking without requiring user accounts. Phase 1 avoids backend complexity. Plan to migrate to server-side storage in Phase 2.

Decision: Use browser LocalStorage for all progress data (sessions, badges, streaks).

Alternatives Considered:
1. **Cookies**: Limited to 4KB, sent with every request (performance overhead)
2. **IndexedDB**: More storage (50MB+), but overkill for simple key-value data
3. **Server-Side Storage**: Best UX, but requires auth system and backend infrastructure

Consequences:
- ✅ Benefits: Zero backend cost, instant reads/writes, works offline, privacy-friendly (no server tracking)
- ❌ Trade-offs: Data lost if user clears browser storage, no cross-device sync, 5-10MB browser limit
- ⚠️ Risks: Users losing progress will be frustrated (mitigate with export feature)

Migration Plan (Phase 2):
1. Add “Export Progress” button (download JSON)
2. Build Cloudflare Workers API for progress storage
3. Implement hybrid system: LocalStorage as cache, server as source of truth
4. Add “Import Progress” for manual data migration

Related Decisions: [DECISION-011]

🎨 Design Decisions

[DECISION-005] Neurodiversity-Affirming Design System

Date: 2024-12-16
Status: Accepted
Context: Target audience includes users with sensory sensitivities, ADHD, autism, dyslexia. Need design system that reduces cognitive load and avoids sensory overwhelm.

Decision: Implement low-stimulation design system with calm colors, ample whitespace, subtle animations, and high contrast.

Design Principles:
1. **Sensory Safety**: No flashing animations, no auto-playing videos, subtle transitions (<300ms)

2. **Clear Hierarchy**: Large headings (2.5rem+), 1.5rem+ body text, scannable layouts
3. **Consistent Patterns**: Reusable components, predictable navigation, minimal surprises
4. **Accessibility First**: 4.5:1 contrast ratios, keyboard navigation, focus indicators

Color Palette:
- Primary (Calm Blue): `#4A90E2` — trust, focus
- Secondary (Soft Green): `#50C878` — success, growth
- Accent (Warm Orange): `#FFB347` — energy, motivation (used sparingly)
- Neutral (Light Gray): `#F5F5F5` — backgrounds
- Text (Dark Gray): `#2C3E50` — readability

Typography:
- Font Family: Inter (sans-serif, dyslexia-friendly alternative to Comic Sans)
- Line Height: 1.6 (enhanced readability)
- Letter Spacing: 0.02em (slight increase for dyslexic users)

Consequences:
- ✅ Benefits: Positive user feedback on “calm” feel, lower bounce rates
- ❌ Trade-offs: Less visually “exciting” than mainstream apps (intentional)
- ⚠️ Risks: May not appeal to neurotypical users seeking “fun” design

Related Decisions: [DECISION-006]

[DECISION-006] Tailwind CSS for Styling

Date: 2024-12-15
Status: Accepted
Context: Needed utility-first CSS framework for rapid prototyping and consistent spacing/colors. Considered Tailwind, CSS Modules, Styled Components.

Decision: Use Tailwind CSS 3.3 with custom configuration for neuro-inclusive design system.

Alternatives Considered:
1. **CSS Modules**: More explicit, but verbose and harder to maintain consistency
2. **Styled Components**: Popular for React, but runtime CSS-in-JS impacts performance
3. **Vanilla CSS**: Full control, but reinventing the wheel for common patterns

Consequences:
- ✅ Benefits: Rapid development, consistent spacing (4px scale), purged unused CSS (<50kB)
- ❌ Trade-offs: HTML becomes verbose with multiple utility classes, learning curve for team
- ⚠️ Risks: Over-reliance on utilities can lead to inconsistent custom styles

Custom Configuration:

```
// tailwind.config.js
theme: {
  extend: {
    colors: {
      primary: '#4A90E2',
      secondary: '#50C878',
      accent: '#FFB347',
    },
    spacing: {
      // 4px base scale
    },
    animation: {
      // Subtle, <300ms transitions
    },
  },
}
```

Related Decisions: [DECISION-005]

[DECISION-007] Radix UI for Accessible Components

Date: 2024-12-17
Status: Accepted
Context: Need accessible, unstyled component primitives (modals, dropdowns, tooltips) that work with screen readers. Considered Radix UI, Headless UI, Reach UI.

Decision: Use Radix UI for all interactive components requiring accessibility attributes.

Alternatives Considered:
1. **Headless UI**: Good for Tailwind, but less comprehensive than Radix
2. **Reach UI**: Excellent accessibility, but opinionated styling and larger bundle
3. **Custom Components**: Full control, but reinventing accessibility is error-prone

Consequences:
- ✅ Benefits: WCAG 2.1 compliant out-of-box, keyboard navigation, focus management, 98+ Lighthouse accessibility scores
- ❌ Trade-offs: Larger bundle size than custom components (~30kB for all primitives)
- ⚠️ Risks: Breaking changes in future Radix versions

Components Used:
- Dialog (modals)
- Dropdown Menu (navigation)
- Tooltip (guidance)
- Accordion (FAQ)
- Progress (session tracking)

Related Decisions: [DECISION-005]

📦 Feature Decisions

[DECISION-008] Voice Guidance: Pre-Recorded Audio + TTS Hybrid

Date: 2024-12-24
Status: Accepted
Context: Breathing techniques need voice coaching for best results. TTS quality varies by browser/device. Pre-recorded audio ensures consistency but lacks flexibility.

Decision: Implement 3-mode voice system: Pre-recorded audio (default), TTS (fallback), Off (user preference).

Alternatives Considered:
1. **TTS Only**: No audio file bandwidth, but quality inconsistent (robotic on some browsers)
2. **Pre-Recorded Only**: Best quality, but inflexible (can't adjust speed, limited language support)
3. **No Voice Guidance**: Simpler implementation, but less effective for breathing technique adherence

Consequences:
- ✅ Benefits: High-quality default experience, works offline (cached audio), user choice
- ❌ Trade-offs: 4 audio files × 1.4 MB = 5.6 MB bandwidth (mitigate with compression + lazy loading)
- ⚠️ Risks: Audio files may not sync perfectly with visual timer (mitigate with precise timing logic)

Implementation Details:
- **Pre-Recorded Audio**: 60-second MP3 files at 128kbps, British female voice (Dorothy)
- **TTS Fallback**: Web Speech API with `rate: 0.9`, `pitch: 1.0` for calm tone
- **Audio Files**:
- `box-breathing-instructions.mp3` (4-4-4-4 pattern)
- `4-7-8-instructions.mp3` (4-7-8 pattern)
- `coherent-instructions.mp3` (5-5 pattern)
- `sos-instructions.mp3` (4-6 pattern)

Related Decisions: [DECISION-009]

[DECISION-009] Web Audio API for Ambient Sounds

Date: 2024-12-24
Status: Accepted
Context: Ambient sounds (rain, ocean, forest) enhance relaxation during breathing sessions. Pre-recorded loops require large audio files and can have noticeable seams.

Decision: Generate ambient sounds procedurally using Web Audio API instead of pre-recorded loops.

Alternatives Considered:
1. **Pre-Recorded Loops**: High-quality, but large file sizes (5-10 MB each) and loop seams
2. **Third-Party API**: Services like Freesound, but requires internet and API limits
3. **Silence**: Simplest, but less immersive experience

Consequences:
- ✅ Benefits: Zero bandwidth (0 MB vs 50+ MB for 7 sounds), seamless loops, customizable (volume, pitch)
- ❌ Trade-offs: More complex code, quality depends on algorithm sophistication
- ⚠️ Risks: May sound "synthetic" compared to real recordings (mitigate with multi-layered oscillators)

Implementation Details:

```
// Rain: White noise + bandpass filter
const rain = audioContext.createBufferSource();
const filter = audioContext.createBiquadFilter();
filter.type = 'bandpass';
filter.frequency.value = 1000; // Hz

// Ocean: Low-frequency oscillation + noise
const ocean = audioContext.createOscillator();
ocean.frequency.value = 0.3; // Hz (slow waves)

// 7 sounds total: Rain, Ocean, Forest, Fire, Singing Bowl, Wind Chimes, None
```

Related Decisions: [DECISION-008]

[DECISION-010] No Third-Party Analytics (Privacy-First)

Date: 2024-12-16
Status: Accepted
Context: Many analytics tools (Google Analytics, Mixpanel) track users across sites, violate GDPR without explicit consent, and contradict our privacy-first principles.

Decision: Ship Phase 1 with no analytics. Implement privacy-friendly alternative (Plausible, Fathom) in Phase 2 only if essential for product decisions.

Alternatives Considered:
1. **Google Analytics**: Free, comprehensive, but privacy-invasive and slow (adds 17kB JS)
2. **Plausible/Fathom**: Privacy-friendly, lightweight (<1kB), but $9-19/month cost
3. **Self-Hosted Plausible**: Best of both worlds, but requires server maintenance
4. **No Analytics**: Simplest, most privacy-respecting, but limits data-driven decisions

Consequences:
- ✅ Benefits: Faster page loads (no tracking scripts), zero privacy concerns, builds user trust
- ❌ Trade-offs: Limited visibility into user behavior (can't A/B test, optimize funnels)
- ⚠️ Risks: May miss critical usability issues without data

Mitigation Strategies:
1. User feedback forms (qualitative data)
2. Browser console error tracking (technical issues)
3. Server logs analysis (page views, referrers)
4. Optional "Share Feedback" button on every page

Phase 2 Criteria for Adding Analytics:
- User base >1,000 WAU (justifies cost)
- Specific product questions needing data (e.g., "Which breathing technique is most popular?")
- Privacy-friendly tool selected (Plausible/Fathom)
- Clear consent mechanism (cookie banner)

Related Decisions: [DECISION-004]

[DECISION-011] Ethical Gamification with Grace Days

Date: 2024-12-19
Status: Accepted
Context: Gamification (streaks, badges) increases engagement but can create anxiety (“I broke my streak!”) and manipulative pressure. Need balance between motivation and mental health.

Decision: Implement “grace day” system where missing 1-2 days doesn’t break streaks, and restarts are celebrated (not shamed).

Alternatives Considered:
1. **Strict Streaks**: Miss one day = reset to 0 (maximizes engagement, but can demotivate)
2. **No Streaks**: Avoids anxiety, but loses motivational power
3. **Freeze Days**: User can “freeze” streak 3x/month (feels transactional)
4. **Grace Days**: Automatic forgiveness for 1-2 missed days (balances engagement + compassion)

Consequences:
- ✅ Benefits: Reduces “all-or-nothing” thinking, supports users with ADHD/depression (executive dysfunction)
- ❌ Trade-offs: Slightly lower daily engagement (users may “plan” to use grace days)
- ⚠️ Risks: Users may abuse grace days (mitigate with 2-day limit)

Implementation Rules:

```
Streak Logic:
- Daily goal: 1 session/day (any technique, any duration)
- Grace days: 2 non-consecutive days/week
- Restart message: "Welcome back! Your journey continues" (not "You failed")
- Lifetime progress: Never resets (e.g., "You've completed 47 sessions total")
```

Related Decisions: [DECISION-004]

🔧 Technical Implementation Decisions

[DECISION-012] Yarn as Package Manager

Date: 2024-12-15
Status: Accepted
Context: Need deterministic dependency resolution and faster installs than npm. Considered npm, Yarn, pnpm.

Decision: Use Yarn 1.22 (Classic) as package manager.

Alternatives Considered:
1. **npm**: Default choice, but slower installs and less deterministic lockfiles
2. **Yarn Berry (v2+)**: Modern features (Plug’n’Play), but breaking changes and ecosystem compatibility issues
3. **pnpm**: Fastest installs, disk-efficient, but less ecosystem adoption

Consequences:
- ✅ Benefits: `yarn.lock` ensures consistent installs across environments, 30% faster than npm

- ❌ Trade-offs: Yarn Classic in maintenance mode (no major updates), some packages prefer npm
- ⚠️ Risks: May need to migrate to pnpm or npm in future

Related Decisions: [DECISION-001]

[DECISION-013] TypeScript Strict Mode

Date: 2024-12-15
Status: Accepted
Context: Need type safety to prevent runtime errors, especially for complex state management (breathing timers, progress tracking).

Decision: Enable TypeScript strict mode for all files.

Configuration:

```
// tsconfig.json
{
  "compilerOptions": {
    "strict": true,
    "noUncheckedIndexedAccess": true,
    "noImplicitReturns": true,
    "noFallthroughCasesInSwitch": true
  }
}
```

Consequences:
- ✅ Benefits: Catches bugs at compile-time, better IDE autocomplete, easier refactoring
- ❌ Trade-offs: More verbose code (explicit types required), longer development time initially
- ⚠️ Risks: Third-party libraries without TypeScript types require `@types/` packages

Related Decisions: [DECISION-001]

[DECISION-014] No Database in Phase 1 (LocalStorage Only)

Date: 2024-12-16
Status: Accepted (Temporary)
Context: Setting up database (Postgres, MySQL) requires backend infrastructure, adds complexity, and increases costs. Phase 1 can function with LocalStorage.

Decision: Defer database implementation to Phase 2. Use LocalStorage for all user data in Phase 1.

Consequences:
- ✅ Benefits: Faster MVP launch, zero hosting costs, works offline
- ❌ Trade-offs: No cross-device sync, data lost if browser cleared
- ⚠️ Risks: User frustration with data loss (mitigate with export feature)

Phase 2 Migration Plan:
1. Choose database: Cloudflare D1 (SQLite-compatible, serverless) or Supabase (Postgres, managed)
2. Build Cloudflare Workers API for CRUD operations
3. Implement hybrid sync: LocalStorage as cache, database as source of truth
4. Add authentication (NextAuth.js)

Related Decisions: [DECISION-004], [DECISION-011]

🚦 Decisions Under Review

[DECISION-015] Flutter vs React Native for Mobile App

Date: 2024-12-25
Status: Under Review
Context: Need native mobile app for iOS/Android in Phase 3. Considering Flutter (Dart) vs React Native (JavaScript).

Options:
1. **Flutter**: Single codebase, high performance, but learning curve (Dart language)
2. **React Native**: Leverage existing React knowledge, but performance issues on complex animations
3. **Native (Swift/Kotlin)**: Best performance, but maintaining 2 codebases

Decision Pending: Prototype both, evaluate in Q1 2025 based on:
- Developer experience
- Performance (breathing timer accuracy)
- Bundle size
- Ecosystem maturity (audio/speech APIs)

Related Decisions: [DECISION-003]

🔄 Superseded Decisions

[DECISION-016] TTS-Only Voice Guidance (Superseded by DECISION-008)

Date: 2024-12-20
Status: Superseded
Original Decision: Use browser TTS only for voice guidance.

Why Superseded: TTS quality inconsistent across browsers (Chrome good, Firefox robotic). User feedback requested “more natural” voice.

New Decision: Hybrid approach with pre-recorded audio as default, TTS as fallback.

📝 Decision-Making Process

How We Make Decisions

1. **Identify Problem**: Document the context and constraints
2. **Research Options**: List 3-5 alternatives with pros/cons
3. **Prototype** (if needed): Build small POC to test viability
4. **Evaluate Trade-offs**: Consider impact on users, performance, maintainability, cost
5. **Document Decision**: Add to this log with rationale
6. **Review Periodically**: Revisit decisions every quarter (some may need revision)

Decision Principles

1. **User-Centric**: Prioritize user benefit over developer convenience
2. **Evidence-Informed**: Base on data, research, user feedback (not assumptions)
3. **Reversible When Possible**: Prefer decisions that can be changed later (avoid lock-in)
4. **Transparent**: Document rationale so future team understands “why”

Document Maintenance
- Review quarterly (March, June, September, December)
- Mark superseded decisions clearly
- Link related decisions for context
- Update when implementing breaking changes