ENCONTRO: VR Encounter Space

Architecture & Development Plan

A multiplayer WebXR experience for soul encounters — forked from Ontik

---

1. PROJECT IDENTITY

Name: Encontro (Portuguese: "encounter") **Tagline**: Soul encounters in virtual space **Design Target**: The phenomenology of light MDMA — reduced threat detection, amplified warmth perception, ego softening, ecstatic connection **Parent Project**: Ontik VR Shader Experience **Relationship**: Sibling project sharing shader infrastructure; Ontik = solo contemplative journeys, Encontro = multiplayer encounter ceremonies

---

2. EXISTING ONTIK INFRASTRUCTURE (What You Already Have)

```
Ontik-vr-shader-experience/
├── .github/workflows/      # GitHub Actions CI/CD → GitHub Pages
├── client/                  # Frontend WebXR application
│   └── src/                 # Three.js + GLSL shaders + WebXR
├── server/                  # Backend (Express/Node likely)
├── shared/                  # Shared types/utilities
├── docs/                    # GitHub Pages deployment
├── CLAUDE_DEV_PRACTICES.md  # Claude Code integration notes
├── CLAUDE_NOTES.md          # Dev context for AI assistance
├── STYLE_PALETTE.md         # Visual design language
├── vite.config.ts           # Vite build system
├── tsconfig.json            # TypeScript configuration
├── package.json             # Dependencies
├── tailwind.config.cjs      # Tailwind (for UI overlays)
└── eslint.config.js         # Linting
```

What transfers directly to Encontro:

- Three.js rendering pipeline

- GLSL shader infrastructure
- WebXR session management
- Vite build system + TypeScript
- GitHub Actions deployment
- Visual style palette (dark, luminous, cosmic aesthetic)
- Claude dev practices

What Encontro adds:

- WebSocket multiplayer networking
- Presence representation system (particle clouds + ethereal face)
- Spatial audio engine (Web Audio API + Tone.js)
- Session/ceremony orchestration (phase-based timer)
- Merge mechanics (presence interpenetration)
- Simple matchmaking server

---

3. ENCONTRO REPOSITORY STRUCTURE

```
Encontro-vr-encounter/
├── .github/
│   └── workflows/
│       ├── deploy-client.yml    # Build & deploy to GitHub Pages
│       └── deploy-server.yml    # Deploy signaling server
│
├── client/                      # WebXR Frontend
│   ├── public/
│   │   ├── audio/               # Ambient soundscapes, tone samples
│   │   └── textures/            # Face alpha maps, particle sprites
│   │
│   ├── src/
│   │   ├── core/
│   │   │   ├── app.ts                # Main app bootstrap
│   │   │   ├── scene.ts              # Three.js scene setup
│   │   │   ├── renderer.ts           # WebXR renderer config
│   │   │   └── input.ts              # Controller/hand input handling
│   │   │
│   │   ├── ceremony/                 # ← THE CORE EXPERIENCE
```

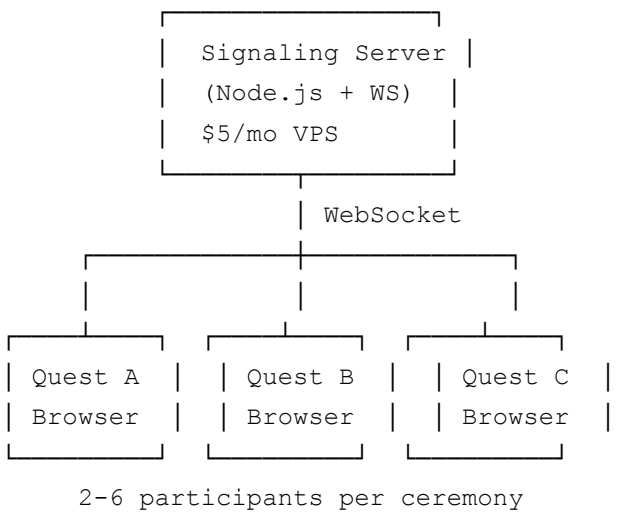
```
│   │   │   ├── CeremonyManager.ts     # Phase orchestration (timer-based)
│   │   │   ├── phases/
│   │   │   │   ├── ArrivalPhase.ts    # Solo settling (3 min)
│   │   │   │   ├── SensingPhase.ts    # Perceiving distant presences (3 min)
│   │   │   │   ├── ApproachPhase.ts   # Moving toward resonance (5 min)
│   │   │   │   ├── EncounterPhase.ts  # Merging & shared experience (10 min)
│   │   │   │   ├── ReleasePhase.ts    # Separation & traces (3 min)
│   │   │   │   └── ReflectionPhase.ts # Solo integration + gift (2 min)
│   │   │   └── CeremonyConfig.ts      # Timing, parameters, phase transitions
│   │   │
│   │   ├── presence/                  # ← LUMINOUS BEING SYSTEM
│   │   │   ├── PresenceManager.ts     # Create/update/destroy presences
│   │   │   ├── LocalPresence.ts       # Your own luminous form
│   │   │   ├── RemotePresence.ts      # Other participants' forms
│   │   │   ├── PresenceRenderer.ts    # GPU particle system for each being
│   │   │   ├── FaceSuggestion.ts      # Ethereal face layer (alpha mesh in parti
│   │   │   ├── MergeSystem.ts         # Presence interpenetration mechanics
│   │   │   └── TraceSystem.ts         # Post-encounter residual effects
│   │   │
│   │   ├── environment/               # ← THE DARK MEADOW
│   │   │   ├── DarkMeadow.ts          # Base environment (bioluminescent ground,
│   │   │   ├── CosmicSky.ts           # Particle-based sky dome
│   │   │   ├── ReactiveGround.ts      # Ground that responds to presence proximi
│   │   │   └── EnvironmentManager.ts  # Phase-responsive environment parameters
│   │   │
│   │   ├── audio/                     # ← SONIC CONNECTION SYSTEM
│   │   │   ├── AudioEngine.ts         # Web Audio API + Tone.js setup
│   │   │   ├── AmbientScape.ts        # Generative ambient soundscape
│   │   │   ├── ToneSignature.ts       # Each user's unique sonic identity
│   │   │   ├── HarmonicResonance.ts   # Proximity-based harmonic blending
│   │   │   ├── SpatialAudio.ts        # 3D positioned audio for presences
│   │   │   └── SharedFrequency.ts     # Merged state sound generation
│   │   │
│   │   ├── shaders/                   # ← YOUR ARTISTIC DOMAIN
│   │   │   ├── presence/
│   │   │   │   ├── presenceVertex.glsl
│   │   │   │   ├── presenceFragment.glsl
│   │   │   │   ├── mergeVertex.glsl
│   │   │   │   └── mergeFragment.glsl
│   │   │   ├── environment/
│   │   │   │   ├── groundVertex.glsl
│   │   │   │   ├── groundFragment.glsl
│   │   │   │   ├── skyVertex.glsl
│   │   │   │   └── skyFragment.glsl
│   │   │   └── post/
│   │   │       ├── bloomPass.glsl
│   │   │       └── colorGrading.glsl
```

```
│   │   │
│   │   ├── network/                    # ← MULTIPLAYER PLUMBING
│   │   │   ├── NetworkManager.ts       # WebSocket connection lifecycle
│   │   │   ├── RoomManager.ts          # Join/leave ceremony rooms
│   │   │   ├── StateSync.ts            # Position/rotation/state interpolation
│   │   │   ├── PresenceSync.ts         # Sync presence visual parameters
│   │   │   └── MessageTypes.ts         # Typed network message definitions
│   │   │
│   │   ├── ui/                         # Minimal pre/post ceremony UI
│   │   │   ├── LobbyUI.ts             # Pre-ceremony waiting space
│   │   │   ├── CeremonyHUD.ts          # Subtle phase indicators
│   │   │   └── PostCeremonyUI.ts       # Reflection prompts
│   │   │
│   │   └── utils/
│   │       ├── math.ts                 # Vector math, interpolation helpers
│   │       ├── easing.ts               # Smooth transitions between phases
│   │       └── biometrics.ts           # Optional: heart rate from controllers
│   │
│   ├── index.html
│   └── vite.config.ts
│
├── server/                             # Signaling & Session Server
│   ├── src/
│   │   ├── index.ts                    # Express + WebSocket server
│   │   ├── CeremonyRoom.ts             # Room state management
│   │   ├── SessionOrchestrator.ts      # Synchronized ceremony timing
│   │   ├── MatchingService.ts          # (Future) AI-assisted proximity shaping
│   │   └── types.ts                    # Shared type definitions
│   ├── package.json
│   └── tsconfig.json
│
├── shared/                             # Shared between client & server
│   ├── CeremonyPhase.ts                # Phase enum + timing constants
│   ├── NetworkMessages.ts              # Message type definitions
│   └── PresenceState.ts                # Presence data structure
│
├── CLAUDE_DEV_PRACTICES.md             # Claude Code working agreements
├── CLAUDE_NOTES.md                     # Project context for AI sessions
├── DESIGN_VISION.md                    # Phenomenological design targets
├── STYLE_PALETTE.md                    # Imported from Ontik + encounter-specific
├── README.md
├── package.json
├── tsconfig.json
└── vite.config.ts
```

---

4. DATA FLOW ARCHITECTURE

Network Topology



State Synchronization (30 Hz)

```
Each client sends per frame:
{
  type: "presence_update",
  userId: string,
  position: { x, y, z },           // Head position
  rotation: { x, y, z, w },        // Head rotation
  leftHand: { pos, rot },          // Controller/hand
  rightHand: { pos, rot },         // Controller/hand
  movementRhythm: number,          // Derived from motion smoothness
  colorState: { h, s, l },         // Current presence color
  breathRate: number,              // If detectable (optional)
  phase: CeremonyPhase,            // Current phase
  mergeTarget: string | null,      // Who they're merging with
  mergeDepth: number               // 0-1 merge intensity
}
```

Bandwidth: ~200 bytes × 30 Hz × 5 peers = ~30 KB/s per user. Trivial.

Merge Detection (Client-Side)

```
When distance(myPresence, otherPresence) < MERGE_THRESHOLD:
  1. Begin visual blending (particle fields overlap)
  2. Send merge_initiate to server
```

```
3. Server confirms mutual merge (both within threshold)
4. Both clients enter merged state
5. Shared visual/audio parameters computed locally
6. Merged state sync at reduced rate (10 Hz)
```

---

5. CEREMONY PHASES — TECHNICAL SPECIFICATION

Phase Timing & Parameters

Phase	Duration	Max Participants Visible	Audio	Interaction
Arrival	3 min	Self only	Solo ambient, generative	Flight, environment response
Sensing	3 min	All (distant, faint)	Distance-attenuated tones	Gravitational pull cues
Approach	5 min	All (closer, brighter)	Harmonic resonance builds	Can move toward others
Encounter	10 min	Full brightness	Merged frequencies	Full merge possible
Release	3 min	Fading	Tones separate	Traces deposited
Reflection	2 min	Self + traces	Solo ambient returns	Gift creation

Phase Transition System

```
interface PhaseConfig {
  duration: number;           // milliseconds
  transitionDuration: number; // fade between phases
  environment: {
    ambientLight: number;     // 0-1
    groundGlow: number;       // 0-1
    skyParticleDensity: number;
    fogDensity: number;
  };
  presence: {
```

```
    visibilityRange: number;  // meters, how far you can see others
    brightnessMultiplier: number;
    mergeEnabled: boolean;
    mergeThreshold: number;   // meters
  };
  audio: {
    ambientVolume: number;
    toneVolume: number;
    harmonicBlending: number; // 0-1, how much tones interact
  };
}
```

Server keeps authoritative ceremony clock. Clients interpolate between phase configs over transitionDuration.

---

6. PRESENCE RENDERING SYSTEM

Luminous Being (GPU Particle Cloud)

Each presence = 2,000-5,000 GPU particles controlled by a custom shader:

```
Core particle behavior:
- Particles orbit a center point (user's tracked position)
- Orbital radius follows Gaussian distribution (dense core, diffuse edges)
- Movement has organic breathing rhythm (sinusoidal scale oscillation)
- Color is HSL-based, shifts slowly over time
- Alpha falls off with distance from center (soft edges)
- Responds to user's movement velocity (faster = more dispersed)
```

Performance budget for Quest standalone:

- **5 presences × 3,000 particles = 15,000 total particles**
- **All GPU-instanced (single draw call per presence)**
- **No physics, no collision — pure visual**
- **Target: 72 Hz sustained on Quest 2, 90 Hz on Quest 3**

Ethereal Face Layer

```
Approach: Low-poly face mesh (200-400 vertices)
- Positioned at head tracking location within particle cloud
- Rendered with extreme alpha (0.05-0.15 opacity)
```

```
- Custom shader: face pixels blend with surrounding particle colors
- Not recognizable as "a face" from distance
- At close range: the impression of human features emerges from light
- Never rendered sharply — always soft, suggested, dreaming

Technical: A single plane with face UV map, or a simple face mesh
with vertex displacement that makes it "breathe" with the particle cloud.
Screen-space effect may work better than geometry.
```

Merge Visual System

```
When two presences overlap:
1. Additive blending of particle systems (colors combine)
2. Particles from each system begin crossing the boundary
   (some of your particles drift into their space, and vice versa)
3. At full merge: unified particle system with combined color palette
4. Post-merge: each presence retains a "trace" — a subtle color
   shift borrowed from the other

Shader approach:
- Each particle has an "owner" attribute (0 or 1)
- In merge state, a "merge_field" uniform (0-1) controls
  how much particles cross the ownership boundary
- Blend factor based on 3D proximity + merge_field
```

---

7. AUDIO SYSTEM

Architecture

```
Web Audio API (low-level) + Tone.js (synthesis)

AudioContext
├── AmbientScape (OscillatorNode chain + ConvolverNode for reverb)
│   └── Generative, evolves with ceremony phase
├── ToneSignature[per user] (Tone.PolySynth)
│   └── Unique frequency set derived from movement patterns
├── HarmonicResonance (Tone.AutoFilter + crossfade)
│   └── Emerges when two signatures interact
├── SpatialAudio (PannerNode per remote presence)
│   └── 3D positioned — hear others from their location
└── MasterGain → destination
```

Tone Signature Generation

Each user's movement rhythm generates a unique harmonic series:

- Base frequency derived from average movement speed (slow = low, active = higher)
- Overtones from movement smoothness (smooth = pure tones, jerky = more harmonics)
- Modulation from hand gestures (reach = pitch bend, circle = tremolo)

When two users approach: their tone signatures begin to interact via ring modulation, creating sum/difference frequencies that neither produces alone. This is the sonic equivalent of "seeing each other."

Spatial Audio

All remote presences use WebAudio PannerNodes:

- HRTF model for natural 3D positioning
- Distance attenuation follows inverse-square with rolloff
- In merge state: audio becomes binaural and centered (you're "inside" each other)

---

8. DEVELOPMENT PHASES & MILESTONES

PHASE 1: Two Beings in Darkness (Weeks 1-8)

Milestone 1.1 — Networked Empty Space (Week 1-2)

- Create repo, fork relevant Ontik infrastructure
- Set up WebSocket server (Node.js, deployable to Railway/Fly.io)
- Implement room creation/joining
- Send/receive position data between two browsers
- Verify: two cubes moving in sync in WebXR

Milestone 1.2 — Luminous Presence (Week 3-4)

- GPU instanced particle system (single presence)
- Particle cloud follows head position with organic motion
- Color/brightness responds to movement

- Soft edges via Gaussian alpha falloff
- Dark meadow environment (ground plane + sky particles)
- Verify: one beautiful luminous being in dark space

Milestone 1.3 — Two Presences See Each Other (Week 4-5)

- Remote presence rendering from network state
- Position interpolation (smooth remote movement)
- Each presence has distinct color
- Distance-based brightness (closer = brighter)
- Verify: two luminous beings seeing each other across dark space

Milestone 1.4 — Approach & Merge (Week 5-7)

- Proximity detection between presences
- Visual blending when within merge threshold
- Particle boundary crossing during merge
- Color transfer (post-merge trace)
- Basic spatial audio (hear the other presence from their location)
- Verify: approach another being, feel the merge, carry their color

Milestone 1.5 — Ceremony Flow (Week 7-8)

- Phase timer system (server-authoritative)
- Environment responds to phase (lighting, fog, particle density)
- Presence visibility changes per phase
- Basic ambient soundscape
- Start-to-finish 25-minute ceremony test
- **FIRST REAL TEST WITH ANOTHER PERSON IN VR**

PHASE 2: Polish & Expand (Weeks 9-16)

Milestone 2.1 — Face Suggestion

- Ethereal face mesh / screen-space face impression

- Alpha tuning (visible only at close range)
- Face “breathes” with particle cloud

Milestone 2.2 — Sound Design

- Tone signature system (movement → unique frequencies)
- Harmonic resonance between approaching presences
- Merge state shared frequencies
- Phase-responsive ambient evolution
- Music/soundscape as connective tissue

Milestone 2.3 — Flight & Ecstasy

- Flight mechanics (controller-based)
- Scale shifts during merge (micro ↔ macro)
- Velocity-responsive visual effects
- “Ecstatic” moments — particle explosions, light surges

Milestone 2.4 — 4-6 Person Ceremonies

- Scale networking to 6 simultaneous users
- Multiple merge pairs possible
- Group merge mechanics (3+ presences?)
- Performance optimization for Quest standalone

PHASE 3: Store & Community (Weeks 17-24)

Milestone 3.1 — PWA Quest Store Packaging

- Progressive Web App manifest
- Bubblewrap packaging for Quest Store
- Store listing, screenshots, description
- Submit for Meta review

Milestone 3.2 — Entrementes Integration

- Scheduled ceremony sessions (weekly/biweekly)

- Pre-ceremony intention setting (text/voice)
- Post-ceremony reflection space
- Facilitator mode (your Cerebro role — you can shape the space)

Milestone 3.3 — Data Collection

- Post-session phenomenological questionnaires
- MEQ30 adaptation for VR encounter
- Communitas scale (from Isness-D)
- Build dataset for research publication

---

9. TECHNOLOGY DECISIONS

Why WebXR First (Not Unity)

Factor	WebXR	Unity
Your existing skills	✅ Three.js, GLSL, TypeScript	❌ C#, Unity editor (learning curve)
Iteration speed	✅ Save → refresh browser	❌ Compile → build → deploy to device
Multiplayer setup	✅ WebSockets (you know this)	⚠️ Netcode/Photon (new tooling)
Quest deployment	✅ PWA via Bubblewrap	✅ Native APK
Performance ceiling	⚠️ Lower (browser overhead)	✅ Higher (native GPU)
Cross-platform	✅ Any WebXR browser	⚠️ Build per platform
Time to first test	✅ 2-3 weeks	❌ 4-6 weeks (setup + learning)

Decision: Build and validate in WebXR. If performance demands native, port to Unity in Phase 3+. The artistic/experiential decisions transfer completely — only the rendering implementation changes.

Server Infrastructure

```
Phase 1-2: Single server
- Railway.app or Fly.io (free tier → $5/mo)
- Node.js + ws (WebSocket library)
- No database needed initially
- Handles: room management, ceremony clock, state relay

Phase 3+: If scaling beyond ~20 concurrent users
- Add Redis for room state
- Consider Cloudflare Workers for WebSocket edge deployment
- Still minimal infrastructure
```

Key Dependencies (Client)

```
{
  "three": "^0.170.0",
  "tone": "^15.0.0",
  "@types/three": "^0.170.0",
  "vite": "^6.0.0",
  "typescript": "^5.7.0"
}
```

Key Dependencies (Server)

```
{
  "ws": "^8.18.0",
  "express": "^4.21.0",
  "typescript": "^5.7.0",
  "tsx": "^4.19.0"
}
```

---

10. CLAUDE CODE WORKFLOW

What Claude Code Handles (Plumbing)

- WebSocket server implementation
- Network message serialization/deserialization
- State interpolation algorithms

- Phase timer system
- Room management logic
- Bubblewrap PWA packaging
- CI/CD pipeline
- Performance profiling scripts
- TypeScript type definitions

What You Handle (Soul)

- Shader aesthetics (how the luminous beings LOOK)
- Color palettes and transitions
- Particle behavior tuning (how presence MOVES)
- Sound design (what the encounter SOUNDS like)
- Ceremony pacing (how the experience FLOWS)
- Face suggestion opacity/style (how human it FEELS)
- Merge feeling (the phenomenological target)
- Environment mood (the dark meadow atmosphere)

Session Pattern with Claude Code

```
1. Open Encontro in Claude Code
2. Describe what you want to build this session
   ("I need the particle system to respond to proximity —
    when two presences get within 3m, their particles
    should start drifting toward each other")
3. Claude Code implements the TypeScript/GLSL
4. You test in browser, adjust visual parameters
5. Commit working state
6. Repeat
```

CLAUDE_NOTES.md Template for Encontro

```
# Encontro - Claude Development Context

## What This Is
```

```
Multiplayer VR encounter ceremony. Users appear as luminous
energy presences in dark space. They can approach, merge, and
separate. Design target: the phenomenology of light MDMA.

## Tech Stack
WebXR + Three.js + TypeScript + WebSocket multiplayer
GPU instanced particles for presence rendering
Web Audio API + Tone.js for spatial/generative audio

## Current Phase
[Update as you go]

## Active Branch
[Current working branch]

## Key Files
- client/src/ceremony/CeremonyManager.ts — phase orchestration
- client/src/presence/PresenceRenderer.ts — particle system
- client/src/shaders/presence/ — GLSL for luminous beings
- server/src/CeremonyRoom.ts — room state

## Design Constraints
- Quest 2 minimum: 72 Hz, 15K particles total budget
- 2-6 participants per ceremony
- 25-minute ceremony arc
- No text chat, no usernames, no profiles
- Connection through presence, sound, light only

## Don't
- Add game mechanics, achievements, scores
- Add social features (friends, follows, profiles)
- Add text or chat
- Optimize prematurely — get it working, then tune
```

---

11. FIRST WEEK ACTION PLAN

Day 1-2: Repository Setup

```
# Create new repo
gh repo create Encontro-vr-encounter --public
cd Encontro-vr-encounter

# Initialize from Ontik's build infrastructure
```

```
# Copy: vite.config.ts, tsconfig.json, eslint.config.js,
#       tailwind.config.cjs, .github/workflows/
# Adapt package.json for Encontro dependencies

# Create directory structure per Section 3
# Add CLAUDE_NOTES.md, DESIGN_VISION.md

# Set up server/ with basic WebSocket echo
# Verify: client connects to server, sends/receives JSON
```

Day 3-4: Two Cubes in Shared Space

```
# Implement:
# - server/src/CeremonyRoom.ts (room join/leave)
# - client/src/network/NetworkManager.ts (WebSocket lifecycle)
# - client/src/network/StateSync.ts (position broadcast)
# - Basic Three.js scene with two cubes

# Verify: Open two browser tabs
# Each tab shows its own cube + the other's cube
# Moving one cube moves it in both tabs
```

Day 5-7: First Luminous Presence

```
# Implement:
# - client/src/presence/PresenceRenderer.ts (GPU particles)
# - client/src/shaders/presence/presenceVertex.glsl
# - client/src/shaders/presence/presenceFragment.glsl
# - Replace cube with particle cloud
# - Basic orbital motion, Gaussian falloff, color

# Verify: One beautiful luminous being floating in dark space
# This is where YOUR eyes matter — tune until it feels right
```

---

12. DESIGN VISION (for DESIGN_VISION.md)

Phenomenological Targets

The space should feel like:

- A clear night in the desert — vast but intimate

- The inside of a cathedral made of darkness
- Being underwater in bioluminescent waters
- The moment before sleep when boundaries soften

Encountering another should feel like:

- Seeing a campfire from across a dark field
- Hearing someone humming a melody you almost recognize
- The warmth of sitting next to someone in comfortable silence
- The moment in MDMA when you look at someone and think “I see you, I actually see you”

Merging should feel like:

- Two candle flames touching
- Harmonizing voices finding each other
- The moment a hug becomes real (when you stop performing it)
- Breathing in sync without trying

After the ceremony:

- Quiet, warm, slightly changed
- Carrying something of the other person
- Like waking from a meaningful dream
- The afterglow

Anti-Targets (What This Is NOT)

- NOT a social platform (no profiles, no followers, no messaging)
- NOT a game (no scores, no achievements, no progression)
- NOT a simulation (no realistic environments, no physics)
- NOT a metaverse (no persistent world, no avatar customization)
- NOT therapy (no facilitator intervention, no processing structure)
- It IS a ceremony space. A place for encounter. Nothing more.

---

13. RISK REGISTER

Risk	Probability	Impact	Mitigation
WebXR performance insufficient on Quest	Medium	High	Profile early; PWA wrapper adds minimal overhead; fall back to fewer particles
Merge mechanic feels gimmicky, not transcendent	Medium	Critical	Test with 2 people ASAP (week 5-6); iterate on feel before features
WebSocket latency makes movement jerky	Low	Medium	Client-side interpolation + prediction; 30 Hz is generous for this use case
Nobody wants to schedule synchronized ceremonies	Medium	Medium	Drop-in sessions (always-on rooms); async presences (recordings)
Scope creep into social platform features	High	High	DESIGN_VISION.md as North Star; no features that aren't about encounter
Daniel's time gets consumed by other projects	High	Critical	Protect 8-10 hrs/week minimum; use Claude Code for all plumbing

---

14. SUCCESS CRITERIA

MVP (Phase 1 Complete)

- Two people have completed a 25-minute ceremony together
- At least one participant reports the experience as "moving" or "meaningful"
- Performance holds at 72 Hz on Quest 2 throughout

v1.0 (Phase 2 Complete)

- 4-6 person ceremonies run smoothly
- Sound design enhances rather than distracts from encounter

- Face suggestion adds humanity without breaking abstraction
- 5 people outside your immediate circle have tried it

Release (Phase 3 Complete)

- Available on Quest Store as PWA
- Monthly Entrementes ceremony sessions running
- Post-session phenomenological data being collected
- You have personally experienced a ceremony that felt like “light MDMA”

This document is the map. The territory is what happens when two luminous beings find each other in the dark.