Preserving Connections: Context-Engineered System Architecture

Technical Specification Document

Version 1.0 | Production-Ready Grief-Tech Platform

Executive Summary

This document outlines a context-engineered system architecture for Preserving Connections, a memory preservation platform that enables families to interact with AI personas of deceased or distant loved ones. The architecture prioritizes emotional safety, context coherence, and horizontal scalability through modular sub-agent design patterns.

Core Engineering Principles:

- Context-first agent design over prompt engineering
- Emotional safety as a first-class system concern
- Horizontal scalability through containerized sub-agents
- Zero-trust security model for sensitive family data
- Spec-driven development with behavioral contracts

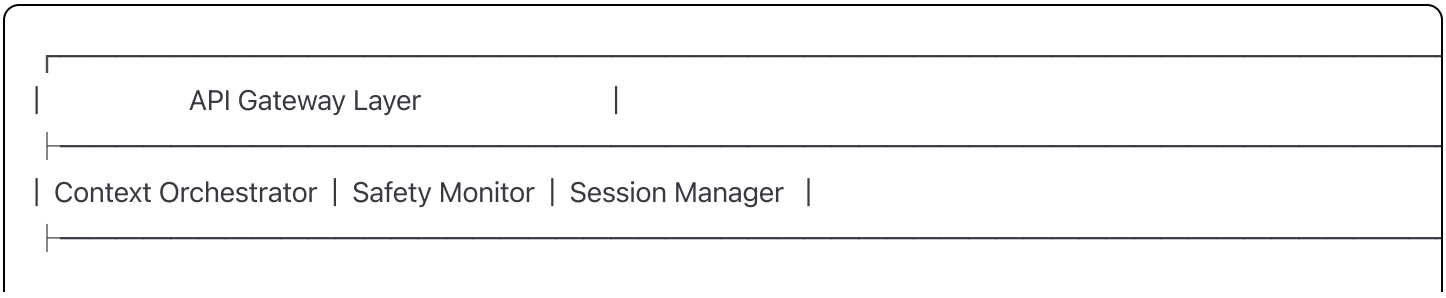
---

1. System Overview

1.1 Architecture Philosophy

The system implements a **Context-Orchestrated Agent Framework (COAF)** where specialized sub-agents manage distinct aspects of persona simulation, memory retrieval, and emotional safety. This approach prevents the brittleness inherent in monolithic prompt-based systems when handling complex, emotionally-sensitive interactions.

1.2 Core Components



```
|  Memory Engine  | Persona Engine | Family Coordinator|
├──────────────────────────────────────────────────────┤
| Vector Store | Relational DB | Media Storage | Cache  |
└──────────────────────────────────────────────────────┘
```

---

2. Sub-Agent Specifications

2.1 Context Orchestrator Agent

Responsibility: Central coordination of all sub-agents and context assembly **Behavioral Contract:**

```typescript
interface ContextOrchestratorSpec {
  assembleContext(userQuery: string, sessionId: string): Promise<ContextFrame>
  routeToAgent(contextFrame: ContextFrame): Promise<AgentResponse>
  compressHistory(conversationHistory: Message[]): Promise<CompressedContext>
  validateResponseSafety(response: string): Promise<SafetyAssessment>
}
```

Context Assembly Rules:

- Maximum 8K tokens per LLM call
- Emotional state awareness in all routing decisions
- Mandatory safety check before response delivery
- Conversation coherence scoring (min 0.8 threshold)

2.2 Memory Curator Agent

Responsibility: Ingestion, categorization, and retrieval of family memories **Behavioral Contract:**

```typescript
interface MemoryCuratorSpec {
  ingestData(rawData: FamilyData, personaId: string): Promise<CategorizedMemory[]>
  retrieveRelevant(query: string, emotionalContext: EmotionalState): Promise<Memory[]>
  compressMemory(oldMemory: Memory, retentionPolicy: Policy): Promise<CompressedMemory>
  tagEmotionalResonance(memory: Memory): Promise<EmotionalTag[]>
}
```

Memory Categorization Schema:

```typescript
type MemoryCategory =
  | 'core_identity'      // Unchanging personality traits
  | 'episodic_life'      // Specific life events
  | 'relationship_dynamics' // Family interaction patterns
  | 'emotional_triggers'    // Topics requiring sensitivity
  | 'wisdom_values'       // Life philosophies, advice
```

2.3 Persona Engine Agent

Responsibility: Maintaining consistent personality simulation **Behavioral Contract:**

```typescript
interface PersonaEngineSpec {
  generateResponse(context: ContextFrame, memories: Memory[]): Promise<PersonaResponse>
  validatePersonaConsistency(response: string, personaProfile: PersonaProfile): Promise<ConsistencyScore>
  adaptToFamilyMember(basePersona: PersonaProfile, familyMember: FamilyMember): Promise<AdaptedPersona
  updatePersonaLearning(interaction: Interaction, feedback: Feedback): Promise<PersonaUpdate>
}
```

Persona Consistency Parameters:

- Voice pattern adherence (sentence structure, vocabulary)
- Emotional boundary respect (topics to approach carefully)
- Relationship-specific adaptation (different tone per family member)
- Cultural/generational context awareness

2.4 Safety Monitor Agent

Responsibility: Real-time emotional safety and intervention protocols **Behavioral Contract:**

```typescript
interface SafetyMonitorSpec {
  assessUserRisk(conversation: Message[], userProfile: UserProfile): Promise<RiskAssessment>
  detectDependencyPatterns(userInteractionHistory: Interaction[]): Promise<DependencyAlert[]>
  flagProfessionalReferral(riskLevel: RiskLevel): Promise<ReferralRecommendation>
  monitorConversationHealth(sessionMetrics: SessionMetrics): Promise<HealthScore>
}
```

Risk Detection Patterns:

- Self-harm ideation language patterns
- Unhealthy dependency on AI persona interactions
- Prolonged grief patterns requiring professional intervention
- Family conflict escalation through shared persona access

2.5 Family Coordinator Agent

Responsibility: Multi-user permissions and family dynamics management **Behavioral Contract:**

```typescript
interface FamilyCoordinatorSpec {
  managePermissions(familyMember: FamilyMember, requestedAccess: AccessRequest): Promise<PermissionRes
  handleConflicts(conflictingMemories: Memory[], stakeholders: FamilyMember[]): Promise<ResolutionPlan>
  coordinateSharing(shareRequest: ShareRequest, recipients: FamilyMember[]): Promise<ShareResult>
  trackConsentChanges(consentUpdate: ConsentUpdate): Promise<ConsentAuditLog>
}
```

---

3. Data Architecture

3.1 Context Compression Strategy

Hierarchical Memory System:

```typescript
```

```
interface ContextHierarchy {
  coreIdentity: {
    personalityTraits: ImmutableTraits
    fundamentalValues: CoreValue[]
    speechPatterns: VoiceSignature
  }
  episodicMemory: {
    recentEvents: TimestampedMemory[]      // Last 30 days, full fidelity
    compressedHistory: SummarizedMemory[]  // Older than 30 days, compressed
    landmarkMoments: SignificantEvent[]    // High emotional weight, preserved
  }
  interactionContext: {
    conversationWindow: Message[]          // Last 20 exchanges
    emotionalState: EmotionalContext       // Current session mood/triggers
    familyMemberContext: RelationshipState // Who's talking, relationship dynamics
  }
}
```

Compression Algorithm:

- Semantic similarity clustering for redundant memories
- Emotional significance weighting (preserve high-impact moments)
- Temporal decay function with landmark event exemptions
- Family consensus integration for disputed memories

3.2 Database Schema Design

Core Tables:

```
sql
```

```sql
-- Persona profiles with vector embeddings for personality consistency
CREATE TABLE persona_profiles (
  id UUID PRIMARY KEY,
  family_id UUID NOT NULL,
  name VARCHAR(255) NOT NULL,
  personality_vector VECTOR(1536),  -- OpenAI embedding dimension
  core_traits JSONB NOT NULL,
  voice_signature JSONB NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Memory fragments with semantic search capability
CREATE TABLE memory_fragments (
  id UUID PRIMARY KEY,
  persona_id UUID REFERENCES persona_profiles(id),
  content_vector VECTOR(1536),
  raw_content TEXT NOT NULL,
  memory_category memory_category_enum NOT NULL,
  emotional_weight FLOAT DEFAULT 0.0,
  compression_level INTEGER DEFAULT 0,  -- 0=original, 1+=compressed
  source_type VARCHAR(50) NOT NULL,
  metadata JSONB,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Family member access control and relationship mapping
CREATE TABLE family_members (
  id UUID PRIMARY KEY,
  family_id UUID NOT NULL,
  email VARCHAR(255) UNIQUE NOT NULL,
  relationship_to_persona VARCHAR(100),
  permission_level permission_enum NOT NULL,
  emotional_boundaries JSONB,  -- Topics to handle carefully
  access_history JSONB,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Conversation sessions with safety monitoring
CREATE TABLE conversation_sessions (
  id UUID PRIMARY KEY,
  family_member_id UUID REFERENCES family_members(id),
  persona_id UUID REFERENCES persona_profiles(id),
```

```
  messages JSONB NOT NULL,
  safety_score FLOAT,
  emotional_health_metrics JSONB,
  ended_at TIMESTAMP WITH TIME ZONE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Safety alerts and professional referral tracking
CREATE TABLE safety_alerts (
  id UUID PRIMARY KEY,
  family_member_id UUID REFERENCES family_members(id),
  alert_type safety_alert_enum NOT NULL,
  risk_level INTEGER NOT NULL,  -- 1-10 scale
  context_data JSONB NOT NULL,
  professional_referral_made BOOLEAN DEFAULT FALSE,
  resolution_status VARCHAR(50) DEFAULT 'active',
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

3.3 Vector Storage Strategy

Memory Retrieval Pipeline:

1. **Semantic Search**: Query embedding → similarity search in memory vectors
2. **Contextual Filtering**: Filter results by emotional appropriateness
3. **Recency Weighting**: Boost recent memories unless landmark events present
4. **Relationship Adaptation**: Adjust memory presentation based on family member relationship

---

4. Security & Privacy Architecture

4.1 Zero-Trust Data Model

Encryption Layers:

- **At Rest**: AES-256 encryption for all stored data
- **In Transit**: TLS 1.3 for all API communications
- **In Memory**: Ephemeral key management for active sessions
- **Vector Embeddings**: Encrypted vector search capability

Access Control Matrix:

```typescript
interface AccessControl {
  familyMemberRoles: {
    'primary_creator': ['full_access', 'invite_others', 'delete_memories']
    'invited_family': ['view_shared', 'add_memories', 'limited_persona_access']
    'posthumous_recipient': ['receive_only', 'no_modification']
    'professional_support': ['safety_monitoring_only']
  }
  memoryVisibility: {
    'public_family': string[]     // Visible to all family members
    'private_relationship': string[] // Only visible to specific relationships
    'creator_only': string[]      // Only visible to memory creator
  }
}
```

4.2 Audit Trail System

Complete interaction logging for:

- Memory access patterns (who accessed what, when)
- Persona response generation (context used, safety checks applied)
- Permission changes (consent modifications, access grants/revocations)
- Safety interventions (alerts triggered, referrals made)

---

5. Replit Implementation Strategy

5.1 Project Structure

```
preserving-connections/
├── src/
│   ├── agents/              # Sub-agent implementations
│   │   ├── context-orchestrator.ts
│   │   ├── memory-curator.ts
│   │   ├── persona-engine.ts
│   │   ├── safety-monitor.ts
│   │   └── family-coordinator.ts
│   ├── core/
│   │   ├── context-frame.ts    # Context management data structures
│   │   ├── compression.ts      # Memory compression algorithms
│   │   └── safety-protocols.ts # Risk assessment logic
```

```
│   ├── data/
│   │   ├── database.ts        # Supabase integration
│   │   ├── vector-store.ts    # Pinecone/Supabase Vector integration
│   │   └── media-storage.ts   # Cloudflare R2 integration
│   ├── api/
│   │   ├── routes/
│   │   └── middleware/
│   └── web/
│       ├── components/
│       ├── pages/
│       └── hooks/
├── specs/                     # Behavioral specifications
│   ├── agent-contracts.ts     # TypeScript interfaces for all agents
│   ├── safety-protocols.md    # Human-readable safety specifications
│   └── persona-consistency.md # Personality maintenance guidelines
├── tests/
│   ├── agent-behavior/        # Sub-agent behavioral testing
│   ├── safety-scenarios/      # Crisis intervention testing
│   └── integration/           # End-to-end flow testing
└── docs/
    ├── onboarding-flow.md     # Family onboarding specifications
    ├── b2b-integration.md     # Funeral home partnership specs
    └── scaling-playbook.md    # Horizontal scaling guidelines
```

5.2 Technology Stack Selection

Backend Framework: Next.js 14 with App Router (TypeScript)

- Server-side rendering for SEO optimization
- Built-in API routes for microservice architecture
- Edge runtime compatibility for global deployment

Database: Supabase (PostgreSQL + Vector Extensions)

- Native vector similarity search
- Row-level security for family data isolation
- Real-time subscriptions for multi-user coordination

LLM Integration: Multi-provider with failover

```typescript
```

```
interface LLMProvider {
  primary: 'openai'    // GPT-4o for persona generation
  secondary: 'anthropic' // Claude for safety monitoring
  tertiary: 'google'   // Gemini for memory categorization
}
```

Caching Strategy: Redis for session state

- Conversation context caching (TTL: 1 hour)
- Memory retrieval result caching (TTL: 24 hours)
- Safety assessment caching (TTL: 5 minutes)

5.3 Development Phases

Phase 1 (Days 1-14): Core Context Framework

```
bash

# Initialize with behavioral specifications first
npx create-next-app@latest preserving-connections --typescript
cd preserving-connections

# Install core dependencies
npm install @supabase/supabase-js openai anthropic @pinecone-database/pinecone
npm install @types/node prisma redis bullmq
npm install -D @types/jest jest ts-jest

# Set up behavioral specification validation
npm install joi zod class-validator
```

Key Deliverables:

- All sub-agent TypeScript interfaces defined
- Context orchestrator with basic routing logic
- Safety monitor with risk pattern detection
- Database schema implementation with vector search

Phase 2 (Days 15-28): Memory & Persona Systems

- Memory curator with semantic categorization
- Persona engine with consistency validation

- Family coordinator with permission management
- Basic web interface for testing

Phase 3 (Days 29-42): Integration & Safety

- End-to-end conversation flows
- Professional referral integration
- B2B landing page and partner APIs
- Comprehensive safety scenario testing

Phase 4 (Days 43-60): Scaling & Polish

- Load testing and performance optimization
- Automated persona quality assurance
- White-label deployment templates
- Production monitoring and alerting

5.4 Critical Configuration

Environment Variables:

```bash
# Core Infrastructure
DATABASE_URL="postgresql://..."
SUPABASE_URL="https://..."
SUPABASE_ANON_KEY="..."
REDIS_URL="redis://..."

# LLM API Keys
OPENAI_API_KEY="sk-..."
ANTHROPIC_API_KEY="sk-ant-..."
GOOGLE_AI_API_KEY="..."

# Safety & Monitoring
CRISIS_HOTLINE_API="..."
PROFESSIONAL_REFERRAL_WEBHOOK="..."
SENTRY_DSN="..."

# B2B Integration
STRIPE_SECRET_KEY="sk_live_..."
FUNERAL_PARTNER_WEBHOOK_SECRET="..."
```

Replit Secrets Setup:

```javascript
// Store all sensitive keys in Replit Secrets
const config = {
  database: {
    url: process.env.DATABASE_URL,
    vectorDimensions: 1536
  },
  llm: {
    primary: process.env.OPENAI_API_KEY,
    secondary: process.env.ANTHROPIC_API_KEY,
    maxTokens: 8000,
    temperature: 0.3  // Lower temperature for consistency
  },
  safety: {
    riskThreshold: 0.7,
    professionalReferralEndpoint: process.env.CRISIS_REFERRAL_API,
    monitoringEnabled: true
  }
}
```

---

6. Quality Assurance Framework

6.1 Behavioral Testing Strategy

Agent Behavioral Validation:

```typescript
```

```
describe('Persona Engine Consistency', () => {
  test('maintains voice patterns across conversations', async () => {
    const persona = await createTestPersona({
      voicePattern: { sentenceLength: 'medium', vocabulary: 'colloquial' }
    })

    const responses = await generateMultipleResponses(persona, testQueries)
    const consistency = measureVoiceConsistency(responses)

    expect(consistency.score).toBeGreaterThan(0.85)
    expect(consistency.vocabularyVariance).toBeLessThan(0.2)
  })
})

describe('Safety Monitor Effectiveness', () => {
  test('detects self-harm ideation patterns', async () => {
    const riskMessages = loadTestScenarios('self-harm-patterns')

    for (const message of riskMessages) {
      const assessment = await safetyMonitor.assessUserRisk(message, userContext)
      expect(assessment.riskLevel).toBeGreaterThan(0.7)
      expect(assessment.recommendsProfessionalReferral).toBe(true)
    }
  })
})
```

6.2 Emotional Safety Validation

Crisis Scenario Testing:

- Suicide ideation detection accuracy (>95% sensitivity)
- Unhealthy dependency pattern recognition
- Family conflict mediation effectiveness
- Professional referral pathway validation

---

7. Scalability Considerations

7.1 Horizontal Scaling Strategy

Containerized Sub-Agent Deployment:

```dockerfile
# Each sub-agent runs in isolated containers
FROM node:18-alpine
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production
COPY src/agents/memory-curator.ts ./
EXPOSE 3001
CMD ["node", "memory-curator.js"]
```

Load Balancing Configuration:

- Context orchestrator: 4 instances minimum
- Memory curator: 2 instances (read-heavy workload)
- Safety monitor: 6 instances (critical path)
- Persona engine: 8 instances (compute-intensive)

7.2 Performance Targets

Response Time SLAs:

- Conversation response: <2 seconds (95th percentile)
- Memory retrieval: <500ms (99th percentile)
- Safety assessment: <1 second (99.9th percentile)
- Family coordination: <3 seconds (90th percentile)

Throughput Capacity:

- 10,000 concurrent conversations
- 1M daily memory retrievals
- 100K daily safety assessments
- 50K monthly new persona creations

Implementation Checklist

Week 1-2: Foundation

- [ ] Repository structure with specifications-first approach
- [ ] Sub-agent TypeScript interfaces and behavioral contracts

- [ ] Database schema with vector search capabilities
- [ ] Basic context orchestrator with routing logic
- [ ] Safety monitor with pattern recognition
- [ ] Comprehensive testing framework setup

Week 3-4: Core Systems

- [ ] Memory curator with semantic categorization
- [ ] Persona engine with consistency validation
- [ ] Family coordinator with permission management
- [ ] Context compression algorithms
- [ ] Basic conversation UI for testing

Week 5-6: Integration

- [ ] End-to-end conversation flows
- [ ] Professional referral system integration
- [ ] B2B partner API endpoints
- [ ] Safety scenario comprehensive testing

Week 7-8: Production Ready

- [ ] Load testing and performance optimization
- [ ] Automated quality assurance pipelines
- [ ] White-label deployment capabilities
- [ ] Monitoring, alerting, and error handling
- [ ] Documentation and handoff materials

Critical Success Factors:

1. **Specifications before implementation** - Define agent behavior contracts first
2. **Safety as core architecture** - Not an afterthought, but foundational
3. **Context compression excellence** - This determines scalability limits
4. **Emotional consistency validation** - Automated testing for persona quality
5. **Family dynamics complexity** - Handle permission conflicts gracefully

This architecture prioritizes the emotional well-being of grieving families while providing the technical foundation for massive scale. The context engineering approach ensures consistent, safe interactions even as the system grows to serve thousands of families simultaneously.