Ticket 7: ABC-2825

Status: Complete
Priority: High
Summary: Memory leaks in WebRTC peer connections causing browser crashes after extended sessions
Reporter: Jennifer Walsh (CallCenter Pro)
Assignee: Kevin Liu (Frontend Engineering)
Created Date: May 15, 2025
Updated Date: May 27, 2025

Description

Customer-Reported Symptoms: CallCenter Pro reported that their agents' browsers were crashing after 3-4 hours of continuous Voice Receptionist usage. Chrome tabs would become unresponsive, and eventually the entire browser would freeze requiring a forced restart. The issue primarily affected agents handling high call volumes (50+ calls per day) and was causing significant productivity loss.

Error Messages/Logs:

```
 [2025-05-15 13:45:22] ERROR MemoryManager: Heap allocation failed -
[2025-05-15 13:45:23] WARN GarbageCollector: Unable to free suffici
[2025-05-15 13:45:24] ERROR WebRTCPeerConnection: Failed to create
[2025-05-15 13:45:25] FATAL BrowserProcess: Renderer process crashe
[2025-05-15 13:45:26] INFO SessionManager: Attempting automatic rec
```

Environmental Details: - Browser: Chrome 124.0.6367.91 (64-bit) on Windows 11 - Memory Configuration: 16GB RAM, Chrome process limit 4GB - Session Duration: 6-8 hours continuous usage - Call Volume: 50-80 calls per agent per day - WebRTC Sessions: Up to 15 concurrent peer connections during busy periods - Agent Workstation: Dell OptiPlex 7090, Intel i7-11700

Impact on Business Operations: - Agent productivity decreased 35% due to browser restart downtime - Customer callbacks required when agents experienced crashes mid-call - IT support tickets increased 400% for "browser freezing issues" - Supervisors manually monitoring agent browsers for crash prevention - Overtime costs increased $12,000/week due to lost productivity

Resolution Details: Root Cause Analysis: Memory profiling revealed that WebRTC PeerConnection objects were not being properly garbage collected after call termination. Event listeners, media streams, and ICE candidates were maintaining references that prevented cleanup. The issue was compounded by Safari-style closure retention in callback functions and improper disposal of AudioContext objects.

Applied Fixes: 1. **WebRTC Resource Management:** - Implemented explicit cleanup in `RTCPeerConnection.close()` calls - Added proper disposal of `MediaStreamTrack` objects - Removed all event listeners before connection termination - Introduced connection pool with maximum lifetime limits (2 hours per connection)

1. **Memory Leak Prevention:**
2. Replaced closure-based callbacks with WeakRef patterns
3. Implemented automatic garbage collection triggers after every 10 calls
4. Added memory usage monitoring with automatic session refresh at 80% threshold

5. Deployed Service Worker for background memory optimization

6. **Audio Context Optimization:**

7. Shared single AudioContext across multiple connections
8. Implemented proper `AudioContext.close()` lifecycle management
9. Reduced audio buffer sizes from 4096 to 1024 samples

10. Added automatic AudioContext suspension during idle periods

11. **Browser Resource Monitoring:**

12. Real-time memory usage display for agents
13. Automatic tab refresh recommendation at 3GB memory usage
14. Performance metrics collection for ongoing optimization
15. Browser health checks with proactive warnings

Verification Steps: - 12-hour continuous usage testing with memory profiling: ✅ Memory stable under 2GB - High-volume call simulation (100 calls over 4 hours): ✅ No memory growth detected - Multi-tab stress testing with 5 concurrent agent sessions: ✅ Passed - Customer pilot with 10 agents over 2 weeks: ✅ Zero browser crashes reported - Production deployment monitoring across 150 agents: ✅ 99.8% session stability