Real-Time Call Translation
Updated Implementation Plan

December 13, 2025 → January 1, 2026

Project Code:	25-2-D-5
Team:	Amir Mishayev, Daniel Fraimovich
Advisors:	Dr. Dan Lemberg, Mrs. Elena Kramer
Institution:	Braude College - Software Engineering
Duration:	19 Days (MVP Sprint)
Final Deadline:	January 1, 2026

Project Summary: This updated plan addresses critical gaps identified in the December 13, 2025 codebase audit. The goal is to achieve a functional MVP where users can make real-time translated voice calls. The plan is organized into 6 three-day sprints, prioritizing core functionality over polish features.

1. Current State Assessment

As of December 13, 2025, the project has achieved approximately 55% completion. The infrastructure and scaffolding are largely in place, but critical real-time audio functionality remains unimplemented.

1.1 What Works (Infrastructure ~90%)

- Docker, PostgreSQL, Redis infrastructure fully configured
- FastAPI backend with health checks, CORS, and WebSocket endpoint
- Database models: User, Call, CallParticipant, Contact, VoiceModel, Message
- Flutter app with 70+ files, all screens, navigation, and providers
- WebSocket connection management and message routing logic
- API endpoints for authentication, contacts, calls, and voice

1.2 Critical Gaps Identified

Issue #1: Backend 500 Error (call_service.py:298)

The start_call endpoint fails because `voice_quality_score` can be None when passed to `set_voice_clone_quality()`. Fix: `user.voice_quality_score or 0`

Issue #2: Audio Recording is Mock Only

The `audio_service.dart` file generates fake audio chunks (`List.filled(1600, count % 255)`) instead of capturing real microphone input. This must be replaced with the 'record' package.

Issue #3: Audio Playback Not Implemented

The playAudio() method simply waits 300ms and does nothing. Real audio playback via 'just_audio' package is required.

Issue #4: No Incoming Call Notification

When User A calls User B, User B has no way to know. Missing: GET /api/calls/pending endpoint, 'incoming_call' WebSocket message, IncomingCallScreen in Flutter.

Issue #5: Voice Cloning Not Integrated

The voice_training_service.py references Coqui xTTS, but the project should use Chatterbox Multilingual Voice Cloning.

2. Implementation Sprints

Sprint 1: December 13-15 — Foundation Fixes

Goal: Enable call initiation to work end-to-end

Day 1 (Dec 13): Fix Backend 500 Error

- ■ Fix call_service.py:298 — handle None voice_quality_score
- ■ Add validation for all user fields before participant creation
- ■ Test /api/calls/start endpoint works
- ■ Verify database records are created correctly

Day 2 (Dec 14): Incoming Call Backend

- ■ Create GET /api/calls/pending endpoint
- ■ Add 'incoming_call' message type to WebSocket protocol
- ■ Implement 30-second timeout for unanswered calls
- ■ Add POST /api/calls/{call_id}/accept endpoint
- ■ Add POST /api/calls/{call_id}/reject endpoint

Day 3 (Dec 15): Incoming Call Flutter

- ■ Create IncomingCallScreen with accept/reject buttons
- ■ Listen for 'incoming_call' WebSocket message in app
- ■ Show IncomingCallScreen when message received
- ■ Handle call timeout (auto-reject after 30 seconds)
- ■ Navigate to ActiveCallScreen on accept

Sprint 1 Deliverable: User A calls User B, B sees incoming call, can accept/reject

Sprint 2: December 16-18 — Real Audio Recording

Goal: Capture and transmit real microphone audio

Day 4 (Dec 16): Audio Recording Implementation

- ■ Add 'record' package to pubspec.yaml
- ■ Implement RealAudioService class
- ■ Request microphone permissions
- ■ Configure 16kHz mono PCM recording
- ■ Stream audio chunks every 200ms

Day 5 (Dec 17): Audio Transmission

- ■ Connect RealAudioService to WebSocketService
- ■ Encode audio chunks as base64 for WebSocket

- Add audio message type to WebSocket protocol
- Test audio reaches backend

Day 6 (Dec 18): Audio Playback Implementation

- Add 'just_audio' package to pubspec.yaml
- Create AudioPlaybackService
- Implement audio queue to prevent overlap
- Play received audio chunks
- Handle audio focus and interruptions

Sprint 2 Deliverable: Audio flows: Mic → WebSocket → Backend → WebSocket → Speaker

Sprint 3: December 19-21 — GCP Translation Pipeline

Goal: Complete Speech-to-Text → Translate → Text-to-Speech pipeline

Day 7 (Dec 19): GCP STT Integration

- ■ Configure GCP credentials in backend
- ■ Test gcp_pipeline.py transcribe_audio() function
- ■ Handle streaming vs batch transcription
- ■ Support Hebrew, English, Russian language codes

Day 8 (Dec 20): Translation & TTS

- ■ Test translate_text() function
- ■ Test synthesize_speech() function
- ■ Configure voice selection per language
- ■ Measure latency (target: <500ms)

Day 9 (Dec 21): End-to-End Pipeline

- ■ Connect audio_worker.py to GCP pipeline
- ■ Route translated audio back to correct participants
- ■ Add transcript message to WebSocket
- ■ Display live transcription in Flutter
- ■ Test HE→EN, EN→RU, RU→HE translations

Sprint 3 Deliverable: Speak Hebrew → Hear English (and vice versa)

Sprint 4: December 22-24 — Voice Cloning (Chatterbox)

Goal: Integrate Chatterbox for voice synthesis with cloned voices

Day 10 (Dec 22): Chatterbox Setup

- ■ Replace Coqui xTTS with Chatterbox in requirements
- ■ Update Dockerfile with Chatterbox dependencies
- ■ Create ChatterboxService class
- ■ Test basic voice synthesis

Day 11 (Dec 23): Voice Model Training

- ■ Update voice recording flow (30 seconds)
- ■ Add random text prompts (jokes/facts) per language
- ■ Upload recording to backend
- ■ Train voice model with Chatterbox

Day 12 (Dec 24): Pipeline Integration

- ■ Replace Google TTS with Chatterbox in pipeline

- Implement fallback to Google TTS if no voice model
- Test voice quality
- Measure latency (target: <1s with voice cloning)

Sprint 4 Deliverable: Translations use speaker's cloned voice

Sprint 5: December 25-27 — Multi-Party & Status

Goal: Support 3-4 participants, online/offline status, call history

Day 13 (Dec 25): Multi-Party Calls

- ■ Test 3-party call setup
- ■ Verify N-to-N translation logic
- ■ Handle participant join during active call
- ■ Handle participant leave during active call
- ■ Retry mechanism for unanswered participants

Day 14 (Dec 26): Online/Offline Status

- ■ Verify status_service.py heartbeat logic
- ■ Add real-time status updates to Flutter
- ■ Show online/offline indicator on contacts
- ■ Handle reconnection gracefully

Day 15 (Dec 27): Call History

- ■ Implement call history storage
- ■ Create call history screen (WhatsApp style)
- ■ Show participants, duration, date
- ■ Allow calling from history

Sprint 5 Deliverable: Group calls work, contacts show status, history saved

Sprint 6: December 28-31 — Testing & Polish

Goal: Stable, demo-ready MVP

Day 16 (Dec 28): Integration Testing

- ■ Test 2-party call end-to-end (HE↔EN)
- ■ Test 3-party call end-to-end (HE↔EN↔RU)
- ■ Test voice cloning quality
- ■ Test network interruption handling

Day 17 (Dec 29): Bug Fixes

- ■ Fix critical bugs found in testing
- ■ Improve error handling
- ■ Add user-friendly error messages
- ■ Test on multiple devices

Day 18 (Dec 30): Performance Optimization

- ■ Measure and optimize latency

- Optimize audio chunk size
- Add connection quality indicator
- Memory leak fixes

Day 19 (Dec 31): Final Testing

- Full regression test
- Document known issues
- Prepare demo script
- Verify all critical features work

Sprint 6 Deliverable: Demo-ready MVP for January 1, 2026

3. Priority Matrix

Priority	Features	Days	Status
P0 (Must Have)	Fix 500 error, Incoming call, Real audio, Translation pipeline	1-9	Critical Path
P1 (Should Have)	Voice cloning, Multi-party, Status, Call history	10-15	Enhanced UX
P2 (Nice to Have)	Performance optimization, Polish, Demo prep	16-19	Can Cut if Needed

4. Success Criteria for January 1, 2026

- ■ User A can initiate a call to User B
- ■ User B receives incoming call notification
- ■ Call connects with real microphone audio
- ■ Speech is transcribed accurately (GCP STT)
- ■ Speech is translated between HE/EN/RU
- ■ Translation is spoken via TTS or Chatterbox
- ■ Live transcription displays on screen during call
- ■ Call ends cleanly without errors
- ■ No 500 errors on any API endpoint

5. Risk Mitigation

Risk	Mitigation Strategy
GCP Latency Too High	Use streaming STT instead of batch; reduce chunk size
Chatterbox Integration Fails	Keep Google TTS as fallback for MVP
Audio Quality Issues	Test on multiple devices early; adjust sample rate
Time Overrun	Cut P2 features if needed; focus only on P0 for MVP

6. Technical Specifications

6.1 Backend Changes Required

- Fix: call_service.py:298 — handle None with `or 0`
- Add: GET /api/calls/pending — returns pending calls for user
- Add: POST /api/calls/{call_id}/accept — accept incoming call
- Add: POST /api/calls/{call_id}/reject — reject incoming call
- Add: WebSocket 'incoming_call' message type
- Replace: Coqui xTTS → Chatterbox in voice_training_service.py

6.2 Flutter Changes Required

- Replace: Mock AudioService → RealAudioService with 'record' package
- Add: AudioPlaybackService with 'just_audio' package
- Add: IncomingCallScreen with caller info, accept/reject buttons
- Add: Handle 'incoming_call' WebSocket message type
- Update: ContactsScreen with online/offline indicators
- Add: CallHistoryScreen with WhatsApp-style list

6.3 Dependencies to Add

pubspec.yaml (Flutter):

```
dependencies: record: ^5.0.4 # Real audio recording just_audio: ^0.9.36 # Audio playback
```

requirements.txt (Python):

```
chatterbox-tts>=0.1.0 # Replace coqui-tts
```

6.4 Performance Targets

Metric	Target	Measurement
Translation Latency (no voice clone)	< 500ms	End-to-end speech to audio
Translation Latency (with voice clone)	< 1000ms	Including Chatterbox TTS
Speech Recognition Accuracy	> 85%	Word Error Rate on test samples
Concurrent Participants	2-4 users	Stable multi-party calls

7. Team & Contact Information

Role	Name	Email
Backend Lead	Amir Mishayev	Amir.mishayev@e.braude.ac.il
Frontend Lead	Daniel Fraimovich	Daniel.fraimovich@e.braude.ac.il
Academic Advisor	Dr. Dan Lemberg	lembergdan@braude.ac.il
Academic Advisor	Mrs. Elena Kramer	elenak@braude.ac.il

Daily Standup Questions

1. What did I complete yesterday?
2. What will I do today?
3. What blockers do I have?

Document Information

Created: December 13, 2025

Based on: Codebase audit and original work plan

Status: UPDATED PLAN - Replaces original 10-week plan

Good luck with the sprint! ■