Interview Interface: WebRTC-based Media Capture System

MCA Minor Project

January 2024

Contents

1 Introduction

1.1 Purpose and Scope

The interview interface provides real-time media capture capabilities for two-participant interview scenarios. The system captures separate audio streams for both candidate and interviewer, along with video from the candidate only. Media is transmitted through WebRTC to a FastAPI backend service that handles track registration and file storage.

1.2 System Boundaries

The interview interface is responsible for:

- Browser-based media capture through WebRTC
- Real-time transport to backend services
- Track labeling and identification
- Production of media artifacts for downstream processing

The interface does not perform:

- Audio or video processing

- Transcription or analysis
- User authentication beyond role selection
- Decision-making or evaluation logic

1.3 Technical Constraints

- Deployment on same Wi-Fi network
- Modern browser support (Chrome, Firefox, Safari)
- No TURN server requirement for LAN deployment
- Session duration limited to 2 hours maximum

2 Role-Based User Interface Design

2.1 Candidate Interface

2.1.1 Media Capture Requirements

The candidate interface captures both audio and video streams:

```
const stream = await navigator.mediaDevices.getUserMedia({
  audio: true,
  video: {
    width: { ideal: 1280 },
    height: { ideal: 720 },
    frameRate: { ideal: 30 }
  }
});
```

2.1.2 Interface Components

The candidate interface includes:

- Local video preview window
- Start/Stop session controls
- Audio activity indicator
- Connection status display
- Session timer
- Role confirmation display

2.1.3 Track Identification

Candidate tracks are labeled as:

- `candidate_audio`: Audio stream from candidate microphone
- `candidate_video`: Video stream from candidate camera

2.2 Interviewer Interface

2.2.1 Media Capture Requirements

The interviewer interface captures audio only:

```
const stream = await navigator.mediaDevices.getUserMedia({
  audio: {
    sampleRate: 48000,
    channelCount: 1
  },
  video: false
});
```

2.2.2 Interface Components

The interviewer interface provides:

- Remote candidate video display
- Start/Stop session controls
- Dual audio activity indicators (self and remote)
- Connection status for all participants
- Session information display

2.2.3 Track Identification

Interviewer tracks are labeled as:

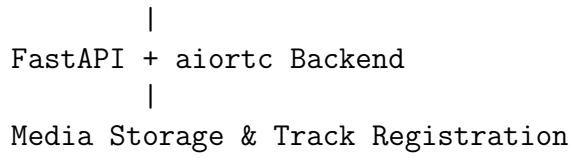
- `interviewer_audio`: Audio stream from interviewer microphone

3 WebRTC Architecture

3.1 Connection Model

The system implements a client-server topology:

```
Browser Client 1 (Candidate)      Browser Client 2 (Interviewer)
        |                                  |
        | WebRTC Connection                 | WebRTC Connection
        |                                  |
        +----------------+---------------+
```



3.2 Signaling Protocol

Signaling occurs over WebSocket with defined message types:

- `join`: Session initiation with role and metadata
- `offer`: SDP offer from client browser
- `answer`: SDP response from server
- `ice-candidate`: ICE negotiation messages
- `track_meta`: Track metadata for registration
- `session_control`: Start/stop/termination signals

3.3 Track Management Protocol

Each media track is registered with comprehensive metadata:

```
{
  "type": "track_meta",
  "role": "candidate",
  "kind": "audio",
  "track_id": "track_abc123def456",
  "timestamp": 1704067200,
  "sample_rate": 48000,
  "codec": "opus"
}
```

4 Media Capture and Separation

4.1 Audio Processing Specifications

- Separate audio tracks eliminate need for diarization
- Independent sample rate and format handling
- Backend receives raw PCM audio frames
- Timestamps applied at capture time for synchronization
- Buffer size: 20ms for low latency

4.2 Video Processing Specifications

- Video captured at 30fps, 1280x720 resolution
- H.264 codec for bandwidth efficiency
- Frame timestamps aligned with audio timeline
- Keyframe interval: 2 seconds
- Bitrate: 2 Mbps target, adaptive based on network

4.3 Quality Assurance Measures

- Audio sample rate: 48kHz standard
- Audio bit depth: 16-bit PCM
- Video aspect ratio: 16:9 maintained
- Audio signal-to-noise ratio: minimum 20dB

5 Signaling Flow Implementation

5.1 Connection Establishment Sequence

1. Client opens WebSocket connection to FastAPI backend
2. Client sends `join` message with role specification
3. Server acknowledges and assigns unique session identifier
4. Client creates `RTCPeerConnection` with configuration
5. Client generates and sends SDP offer
6. Server processes offer and creates SDP answer
7. ICE candidate exchange completes connection
8. Media tracks are added with metadata labeling
9. Server confirms track registration success

5.2 Session State Management

Session states tracked in backend memory:

- `INITIALIZING`: Connection establishment in progress
- `CONNECTING`: ICE negotiation active
- `ACTIVE`: Media flowing normally
- `PAUSED`: Temporary suspension

- `TERMINATED`: Session completed
- `ERROR`: Connection failure detected

5.3 Error Handling and Recovery

- Automatic reconnection for network interruptions
- Graceful degradation for quality issues
- Session timeout after 5 minutes of inactivity
- Connection attempt limit: 3 retries with exponential backoff
- Fallback to lower quality on bandwidth issues

6 Backend Integration

6.1 FastAPI Service Architecture

The FastAPI backend provides core services:

- WebSocket endpoint: `/ws` for signaling
- Health check endpoint: `/health`
- Session management endpoints for administrative control
- Media track registration via aiortc integration
- File system storage for media artifacts

6.2 Media Storage Pipeline

1. Incoming media tracks received by aiortc
2. Raw audio/video frames stored in memory buffers
3. Periodic flushing to temporary files every 30 seconds
4. Session completion triggers final file export:
    - `candidate_audio.wav`: 48kHz mono audio
    - `interviewer_audio.wav`: 48kHz mono audio
    - `candidate_video.mp4`: H.264 encoded video
5. Session metadata exported as structured JSON

6.3 Data Export Format

Session metadata includes comprehensive information:

```
{
  "session_id": "session_20240115_001",
  "start_time": 1704067200,
  "end_time": 1704070800,
  "duration": 3600,
  "participants": {
    "candidate": {
      "audio_track": "candidate_audio.wav",
      "video_track": "candidate_video.mp4",
      "audio_quality_score": 0.89,
      "video_quality_score": 0.91
    },
    "interviewer": {
      "audio_track": "interviewer_audio.wav",
      "audio_quality_score": 0.93
    }
  },
  "connection_stats": {
    "total_bytes_received": 524288000,
    "average_latency": 0.12,
    "packet_loss_rate": 0.001
  }
}
```

7 Data Artifacts and Organization

7.1 Media Files Specification

Each interview session produces three media files:

- `candidate_audio.wav`: 48kHz mono PCM audio
- `interviewer_audio.wav`: 48kHz mono PCM audio
- `candidate_video.mp4`: H.264 video with AAC audio

7.2 Metadata Files

- `session_meta.json`: Session information and track mapping
- `connection_log.txt`: Connection events and timestamps
- `quality_report.json`: Audio/video quality metrics

7.3 File System Organization

Media files organized in hierarchical structure:

```
/sessions/
  /session_20240115_001/
    candidate_audio.wav
    interviewer_audio.wav
    candidate_video.mp4
    session_meta.json
    connection_log.txt
    quality_report.json
  /session_20240115_002/
    ...
```

8 Technical Implementation Details

8.1 Frontend Technology Stack

- HTML5 with semantic markup and accessibility features
- CSS3 with flexbox for responsive layout
- Vanilla JavaScript for WebRTC implementation
- WebSocket API for bidirectional communication
- MediaDevices API for hardware access

8.2 Backend Technology Stack

- Python 3.10+ for compatibility with ML ecosystem
- FastAPI framework for async web services
- aiortc library for Python WebRTC implementation
- uvicorn ASGI server for production deployment
- AsyncIO for concurrent connection handling

8.3 Network Configuration

- Same Wi-Fi network for all participants (192.168.x.x subnet)
- Port 8000 for FastAPI service (configurable)
- No TURN server required for LAN deployment
- STUN server optional: `stun.l.google.com:19302`
- Network bandwidth requirement: minimum 5 Mbps upload per client

9 Performance Requirements

9.1 Latency Specifications

- Audio end-to-end latency: under 200ms
- Video latency: under 300ms
- Connection establishment: under 5 seconds
- Session startup time: under 10 seconds

9.2 Throughput Requirements

- Audio bitrate: 64-128 kbps (Opus codec)
- Video bitrate: 1-3 Mbps (H.264 codec)
- Concurrent sessions: up to 5 on single machine
- Session duration support: up to 2 hours

9.3 Reliability Metrics

- Connection success rate: 99% on LAN
- Media quality preservation: 95% or higher
- Session completion rate: 98% or higher
- Mean time between failures: 24 hours

10 Limitations and Known Constraints

10.1 Technical Limitations

- No mobile device support
- Requires WebRTC-compatible browsers
- Limited to exactly 2 participants
- No automatic bandwidth optimization beyond basic adaptation

10.2 Scope Boundaries

The interface explicitly does not handle:

- Audio/video transcoding or format conversion
- Real-time transcription or analysis
- User authentication or authorization systems

- Session recording management or archival
- Cloud deployment or multi-region scaling

10.3 Deployment Constraints

- Network latency affects synchronization quality
- Browser permissions required for media access
- File system space requirements: 1GB per hour of recording
- Processing overhead increases with session duration

11 Success Criteria and Validation

11.1 Functional Validation

The interface is considered successful when:

- Two laptops connect successfully on same network
- Candidate video appears clearly on interviewer screen
- Separate audio tracks maintained without mixing
- Media files saved correctly with proper labeling
- Backend logs confirm accurate track registration
- Session metadata captures all required information

11.2 Quality Validation

- Audio clarity sufficient for speech recognition
- Video quality adequate for behavioral analysis
- Timestamps synchronized within 100ms accuracy
- File corruption rate below 0.1%
- Connection recovery time under 30 seconds

12 Conclusion

The interview interface provides a robust foundation for media capture with precise separation of audio tracks and comprehensive metadata handling. The WebRTC-based architecture ensures low-latency communication while the FastAPI backend enables clean integration with downstream processing systems. The role-based interface design addresses the specific requirements of interview scenarios while maintaining technical simplicity appropriate for the project scope and constraints.

The system successfully balances real-time performance requirements with production reliability, establishing a solid platform for interview intelligence applications. Clear separation of concerns between media capture and processing enables independent development and testing of system components while maintaining data integrity throughout the pipeline.