P R O J E C T D O C U M E N T

Test Plan / Test Cases Design Document

Project Name	프라이버시 보호 실시간 지원 서비스

12조

202202624 이예인

202002569 최동현

지도교수: 장진수 교수님

Table of Contents

1. Introduction

1.1. Objective

이 문서는 "프라이버시 보호 실시간 지원 서비스" 시스템 중 화면 공유자(이하 요청자) 측의 크롭 기능 및 AI 마스킹 기능에 대한 테스트 계획 및 테스트 케이스 명세를 포함한다.

테스트 계획 문서는 요청자 클라이언트에서 구현된 크롭 및 AI 마스킹 로직이 설계대로 정확하게 비디오 프레임을 처리하고, 개인 정보 보호 기능을 효과적으로 수행하는지 검증하기 위한 전반적인 활동 계획을 포함한다.

테스트 케이스 명세는 요청자의 크롭 및 AI 마스킹 기능과 관련된 UI 상호작용, 내부 처리 로직(Web Worker 포함), 그리고 수정된 비디오 스트림 생성 과정을 구체적으로 테스트하기 위한 케이스들을 기술한다.

2. Test Plan

1. 배경과 목적

1.1 배경

최근 비대면 협업 및 온라인 커뮤니케이션 증가로 화면 공유 기능의 사용이 보편화되었다. 그러나 화면 공유 중 의도치 않게 개인적인 정보나 민감한 데이터(알림 메시지, 바탕화면 아이콘, 특정 애플리케이션 창 등)가 노출될 수 있는 프라이버시 침해 문제가 발생하고 있다.

본 "프라이버시 보호 실시간 지원 서비스"는 이러한 문제를 해결하기 위해 개발되었으며, 요청자가 공유 범위를 직접 지정하는 크롭 기능과 AI를 활용한 자동 마스킹 기능을 통해 화면 공유 환경에서의 사용자 프라이버시를 강화하고자 한다.

중요한 비즈니스 프로세스(사용자 시나리오):

- 요청자가 화면 공유 시작 및 특정 창 또는 전체 화면 선택
- 요청자가 공유 화면의 특정 영역을 선택(크롭)하여 지원자(화면을 보는 사람, 이하 지원자)에게 해당 영역만 전송
- 요청자의 화면에서 AI가 민감 정보(예: 카카오톡 알림)를 실시간으로 감지하여 해당 부분을 마스킹 처리 후 지원자에게 전송
- 지원자는 요청자가 공유하는 (크롭되거나 마스킹된) 화면을 실시간으로 수신

1.2 테스트 목적

- 요청자 클라이언트의 크롭 기능이 사용자의 설정에 따라 정확히 화면 영역을 잘라내는지 확인한다.
- 요청자 클라이언트의 AI 마스킹 기능이 민감 정보(예: 카카오톡 알림창)를 정확히 탐지하고 효과적으로 마스킹 처리하는지 확인한다.
- 위 기능들이 요청자 측에서 처리될 때 성능 저하 없이 안정적으로 동작하는지 검증한다.
- 최종적으로 요청자에서 지원자로 전송될 비디오 스트림이 의도한 대로 수정되었는지 확인한다.

2. 테스트 상세

2.1 테스트 항목
화면 공유자(Receiver) 클라이언트 측의 크롭 처리, AI 마스킹 처라
2.2 테스트될 요소(features)

• 크롭 (Crop) 기능 성능 (요청자 측):
  • 시스템 자원 사용량: 크롭 기능 활성화/비활성화에 따른 요청자 PC의 CPU 및 메모리 사용량 변화량 및 점유율.
  • 스트림 품질: 크롭 기능 사용 중 화면 공유 스트림의 초당 프레임 수(FPS) 변화 (시각적인 부드러움)
• AI 마스킹 (Masking) 기능 성능 (요청자 측):
  • 프레임 처리 시간: AI 모델 추론 및 마스킹 적용에 소요되는 평균 시간 (ms/frame).
  • 시스템 자원 사용량: AI 마스킹 기능 활성화/비활성화에 따른 요청자 PC의 CPU 및 메모리 사용량 변화량 및 점유율 (모델 로드 포함).
  • 스트림 품질: AI 마스킹 기능 사용 중 화면 공유 스트림의 FPS 변화 (시각적인 부드러움).

2.3 테스트되지 않을 요소
• 채팅 기능 • 화상통화 기능 (오디오/비디오 통화) • 로그인 및 사용자 인증 기능 • 크롭/마스킹 UI의 세부적인 디자인 요소, 다양한 예외적 UI 상태에 대한 상세한 사용성 검증 (기본적인 기능 조작 가능 여부만 확인).
2.4 접근 방법
테스트 단계: 1. 기준선 설정   • 요청자 클라이언트에서 크롭 및 AI 마스킹 기능을 모두 비활성화한 상태로 기본적인 화면 공유(정적 콘텐츠, 동적 콘텐츠 각각)를 실행합니다.

- 이때 RTCPeerConnection.getStats() API ()를 주기적으로 호출하여 주요 성능 지표(발신 비트 전송률, 발신 프레임 속도 등)를 수집하여 비교를 위한 기준선으로 삼습니다.

2. 요청자 측 개별 기능 성능 벤치마킹

- 크롭 기능 성능 측정: 크롭 기능만 활성화한 상태에서 다양한 시나리오(예: 다른 크롭 영역)에 따라 성능을 측정합니다.
- AI 마스킹 기능 성능 측정: AI 마스킹 기능만 활성화한 상태에서 다양한 시나리오(예: 다수 알림 발생)에 따라 성능을 측정합니다.
- 각 테스트 시, getStats() API를 주기적으로(예: 1초 간격) 호출하여 요청자의 RTCPeerConnection에서 발신 스트림(outbound-rtp) 및 인코딩 관련 통계, CPU 제한 여부(qualityLimitationReason) 등의 KPI를 수집 및 분석합니다.

테스트 기법: getStats() API 활용을 통한 KPI 수집 및 분석

- 주기적 데이터 수집: RTCPeerConnection.getStats() 메소드를 일정한 간격(예: 1초)으로 호출하여 RTCStatsReport를 확보합니다. 수집된 데이터는 로깅하거나 실시간으로 분석합니다.
- 핵심 KPI 계산 및 모니터링 (요청자 발신 스트림 중심): 발신 비디오 비트 전송률: outbound-rtp 통계의 bytesSent와 timestamp 변화량을 사용하여 계산
  - 발신 비디오 프레임 속도: outbound-rtp 통계의 framesEncoded (또는 framesSent)와 timestamp 변화량을 사용하여 계산합니다.
  - 인코딩 성능 및 CPU 부하 간접 지표: outbound-rtp의 qualityLimitationReason: "cpu" 또는 "bandwidth" 제한 여부를 확인하여 CPU 병목 또는 네트워크 적응 상태를 파악합니다.

2.5 테스트 항목의 pass/fail 기준

(정확한 숫자는 수정될 수 있음)

- 크롭 기능 성능 (요청자 발신 스트림 기준, getStats() 및 OS 모니터링):
  - CPU 부하: outbound-rtp.qualityLimitationReason이 "cpu"로 지속되지 않아야 하며, OS 측정 기준 CPU 사용률 증가폭은 기준선 대비 30% 이내여야 Pass.
  - 메모리 사용량: OS 측정 기준 메모리 사용량 증가폭은 기준선 대비 50MB 이내

여야 Pass.

- 처리 효율: outbound-rtp.framesEncoded로 계산된 평균 FPS가 20fps 이상을 유지하고, outbound-rtp.totalEncodeTime 변화량 / framesEncoded 변화량으로 계산된 프레임당 평균 추가 인코딩 시간이 기준선 대비 10ms 이내 증가여야 Pass.

- AI 마스킹 기능 성능 (요청자 발신 스트림 기준, getStats() 및 OS 모니터링):
  - CPU 부하: outbound-rtp.qualityLimitationReason이 "cpu"로 지속되지 않아야 하며, OS 측정 기준 CPU 사용률 증가폭은 기준선 대비 50% 이내여야 Pass.
  - 메모리 사용량: OS 측정 기준 메모리 사용량 증가폭은 기준선 대비 200MB 이내여야 Pass (AI 모델 크기 고려).
  - 처리 효율: outbound-rtp.framesEncoded로 계산된 평균 FPS가 15fps 이상을 유지하고, outbound-rtp.totalEncodeTime 변화량 / framesEncoded 변화량으로 계산된 프레임당 평균 추가 인코딩 시간이 기준선 대비 40ms 이내 증가여야 Pass.

2.6 테스트 산출물(deliverables)

테스트 종료 후 다음의 산출물을 인도한다.

- 테스트 계획
- 테스트 케이스 명세
- 테스트 결과 보고서

3. 테스트 관리

3.1 작업

- 테스트 계획 수립, 검토 및 최종 확정 (getStats() API 전용 측정 방법론 반영).
- 성능 중심의 테스트 케이스 상세 설계 (getStats() 주요 KPI 측정 항목 명시) 및 검토.
- 성능 측정 및 getStats() 데이터 수집을 위한 요청자 테스트 환경 구축 (테스트 데이터, 측정 도구, getStats() 데이터 로깅 및 분석 스크립트 준비 - 필요시).
- AI 마스킹 성능 테스트를 위한 표준화된 알림 발생 시나리오 및 데이터셋 준비.
- 정의된 테스트 케이스에 따른 성능 테스트 수행 및 getStats() 기반의 정량적 데이터 기록.
- 발견된 성능 문제 및 기능 결함에 대한 상세 보고 및 추적 관리.
- 수정된 사항에 대한 반복적인 성능 검증 (Regression Test).

- 수집된 성능 데이터를 기반으로 결과 분석 및 최종 테스트 보고서 작성.

3.2 기술 자원

- **클라이언트:**
  - 요청자 PC
  - 지원자 PC
- **AI 모델 및 관련 파일:** 텐서 플로우로 학습한 모델AI 모델

3.3 책임과 권한 (인력 자원)

크롭 테스트 담당자 (이예인)

- 테스트 계획 및 테스트 케이스의 구체화, 실행, 결과 기록
- getStats() API를 활용한 성능 데이터의 정확한 수집 및 로깅.
- 결함 수정 후 재검증 테스트수행 및 결과 확인.
- 최종 성능 테스트 결과 보고서 및 전반적인 테스트 결과 보고서 작성

마스킹 테스트 담당자 (최동현)

- 지속적인 모델 테스트와 개선작업 진행
- 결함 수정 후 재검증 테스트수행 및 결과 확인.
- 최종 성능 테스트 결과 보고서 및 전반적인 테스트 결과 보고서 작성 주도.

3.4 훈련

WebRTC getStats() API 심층 이해

- RTCPeerConnection.getStats() API의 반환 값(RTCStatsReport) 구조 및 주요 통계 객체(outbound-rtp, codec 등)의 각 필드 의미
- 특히 outbound-rtp 통계 내 bytesSent, framesEncoded, totalEncodeTime, qualityLimitationReason 등 요청자 측 성능 분석에 핵심적인 지표의 해석 방법 및 활용 방안

3.5 일정

- 테스트 계획 및 케이스 정의 완료: **2025년 5월 17일**
- 테스트 환경 구축 및 기준선(Baseline) 측정 완료: **2025년 5월 18일**

• 크롭 기능 성능 테스트 수행 및 1차 분석: **2025년 5월 19일 ~ 2025년 5월 23일** • AI 마스킹 기능 성능 테스트 수행 및 1차 분석: **2025년 5월 19일 ~ 2025년 5월 23일** • 성능 개선 및 수정: **테스트 수행 후 지속적으로 진행**
3.6 위험 요소와 비상 대처 상황
크롭 또는 AI 마스킹 기능 활성화 시, 요청자 PC의 성능 저하(CPU 과부하, 심각한 FPS 드롭, 메모리 부족 현상 등)가 Pass/Fail 기준을 현저히 초과하여 실사용이 불가능한 수준으로 판단될 경우.

3. Test Cases

1. 서론

1.1 테스트 범위

본 테스트 케이스 명세는 "프라이버시 보호 실시간 지원 서비스"의 요청자 측 화면 공유 크롭(Crop) 기능 및 AI 기반 자동 마스킹(Masking) 기능 각각의 성능을 getStats() API를 활용하여 중점적으로 측정하고 평가한다. 기능의 기본적인 시각적 정확성은 성능 측정의 전제 조건으로 확인합니다. 크롭 기능과 AI 마스킹 기능은 동시에 활성화하여 테스트하지 않는다.

1.2 테스트 상황

모든 테스트는 요청자 클라이언트(예: Chrome 브라우저)에서 수행되며, getStats() API를 통해 성능 데이터를 수집하고 OS 수준에서 CPU/메모리 사용량을 모니터링한다. 지원자 클라이언트는 요청자가 공유하는 화면을 수신하여 기능의 기본적인 시각적 동작을 확인하는 보조적인 역할만 수행한다.

1.3 문서 표기법

각 테스트 케이스는 ID, 테스트 대상(기능), 테스트 조건(상세 절차), 테스트 데이터, 예상 결과(getStats() 기반 주요 KPI 성능 목표치 및 기본적인 기능 확인사항 포함), 실제 결과(측정된 KPI 값), Pass/Fail 여부, 비고(측정 환경, getStats() 주요 확인 항목 등) 항목으로 구성됩니다.

2. 테스트 케이스

2.1 테스트 케이스 명세

ID	테스트 대상 (요청자)	테스트 조건 (절차)	테스트 데이터	예상 결과
기준선 측정				
PERF-	화면 공유 (기능	1. 요청자가 지원자	공유 대상: 텍스트	OS CPU 사용량:

B-001	비활성화, 정적 콘텐츠)	와 WebRTC 연결 후 화면 공유 시작 (공유 대상: 지정된 정적 웹페이지). 2. 화면 공유 안정화 후 1분 대기. 3. 이후 3분 동안 1초 간격으로 getStats() 데이터 수집 및 OS 수준 CPU/메모리 사용량 기록. 4. 수집된 데이터로 평균 CPU/메모리 사용량, outbound-rtp의 평균 발신 비트 전송률 및 framesEncoded 기반 FPS 계산.	및 이미지로 구성된 표준 정적 웹페이지 (스크롤 없음)	X_base % 이하, OS 메모리 사용량: Y_base MB 이하. outbound-rtp: bytesSent 변화량 기반 비트 전송률 안정적, framesEncoded 변화량 기반 FPS 25~30fps 범위 내 안정적. qualityLimitationReason 필드가 "none" 또는 "bandwidth" 값을 나타냄.
PERF-B-002	화면 공유 (기능 비활성화, 동적 콘텐츠)	1. 요청자가 지원자와 WebRTC 연결 후 화면 공유 시작 (공유 대상: 지정된 동영상 재생). 2. 화면 공유 안정화 후 1분 대기. 3. 이후 3분 동안 1초 간격으로 getStats() 데이터 수집 및 OS 수준 CPU/메모리 사용량 기록. 4. 수집된 데이터로	공유 대상: 1080p, 30fps 로컬 동영상 파일 재생 중인 미디어 플레이어 창	OS CPU 사용량: Xd_base % 이하, OS 메모리 사용량: Yd_base MB 이하. outbound-rtp: bytesSent 변화량 기반 비트 전송률 안정적(콘텐츠 특성 따라 변동), framesEncoded 변화량 기반 FPS 25~30fps 범위 내 안정적. qualityLimitationReason 필드가 "none" 또는 "bandwidth" 값을 나타냄.

		평균 CPU/메모리 사용량, outbound-rtp의 평균 발신 비트 전송률 및 framesEncoded 기반 FPS 계산.		
크롭 기능 성능				
PERF-C-001	크롭 기능 (정적 콘텐츠, 화면 중앙 50% 크롭)	1. PERF-B-001과 동일한 환경에서 크롭 기능 UI를 통해 화면 중앙 50% 영역으로 크롭 설정 후 "적용". 2. 크롭 적용 안정화 후 1분 대기. 3. 이후 3분 동안 getStats() 데이터 및 OS 수준 CPU/메모리 측정. 4. (지원자 화면에서 크롭된 영역이 시각적으로 정확한지 간략히 확인)	크롭 설정: 화면의 Top 25%, Bottom 25%, Left 25%, Right 25% 제외 (중앙 50%만 공유)	OS CPU 사용량: X_base % 대비 15%p 이내 증가. OS 메모리 사용량: Y_base MB 대비 50MB 이내 증가. outbound-rtp.framesEncoded FPS 20fps 이상 유지. outbound-rtp.totalEncodeTime 증가량 / framesEncoded 증가량 (프레임당 평균 추가 인코딩 시간)이 기준선 대비 10ms 이내. qualityLimitationReason이 "cpu"가 아님. 지원자 화면에서 크롭 정상 확인.
PERF-C-002	크롭 기능 (동적 콘텐츠, 화면 상단 25% 크롭)	1. PERF-B-002와 동일한 환경에서 크롭 기능 UI를 통해 화면 상단 25% 영역으로 크롭 설정 후 "적용".	크롭 설정: 화면의 Top 0%, Bottom 75% 제외 (상단 25%만 공유)	OS CPU 사용량: Xd_base % 대비 15%p 이내 증가. OS 메모리 사용량: Yd_base MB 대비 50MB 이내 증가. outbound-

		2. 크롭 적용 안정화 후 1분 대기. 3. 이후 3분 동안 getStats() 데이터 및 OS 수준 CPU/메모리 측정. 4. (지원자 화면에서 크롭된 영역이 시각적으로 정확한지 간략히 확인)		rtp.framesEncoded FPS 20fps 이상 유지. outbound-rtp.totalEncodeTime 증가량 / framesEncoded 증가량 기준선 대비 10ms 이내. qualityLimitationReason이 "cpu"가 아님. 지원자 화면에서 크롭 정상 확인.
AI 마스킹 기능 성능				
PERF-M-001	AI 마스킹 (정적 콘텐츠, 카카오톡 알림 1개)	1. PERF-B-001과 동일한 환경에서 AI 마스킹 기능 활성화. 2. 화면의 특정 위치에 표준 카카오톡 알림 1개 발생 및 유지. 3. 마스킹 적용 안정화 후 1분 대기. 4. 이후 3분 동안 getStats() 데이터 및 OS 수준 CPU/메모리 측정. 5. (지원자 화면에서 알림이 마스킹 처리되었는지 시각적으로 간략히 확	테스트용 카카오톡 알림 이미지 또는 스크립트로 1개 알림 발생 (고정된 위치, 일반 텍스트 메시지)	OS CPU 사용량: X_base % 대비 30%p 이내 증가. OS 메모리 사용량: Y_base MB 대비 200MB 이내 증가. outbound-rtp.framesEncoded FPS 15fps 이상 유지. outbound-rtp.totalEncodeTime 증가량 / framesEncoded 증가량 기준선 대비 40ms 이내. qualityLimitationReason이 "cpu"가 아님. 지원자 화면에서 알림 마스킹 정상 확인.

		인)		
PERF-M-002	AI 마스킹 (정적 콘텐츠, 카카오톡 알림 3개)	1. PERF-B-001과 동일한 환경에서 AI 마스킹 기능 활성화. 2. 화면의 서로 다른 위치에 표준 카카오톡 알림 3개 동시 발생 및 유지. 3. 마스킹 적용 안정화 후 1분 대기. 4. 이후 3분 동안 getStats() 데이터 및 OS 수준 CPU/메모리 측정. 5. (지원자 화면에서 모든 알림이 마스킹 처리되었는지 시각적으로 간략히 확인)	테스트용 카카오톡 알림 이미지 또는 스크립트로 3개 알림 동시 발생 (고정된 서로 다른 위치)	OS CPU 사용량: X_base % 대비 35%p 이내 증가. OS 메모리 사용량: Y_base MB 대비 220MB 이내 증가. outbound-rtp.framesEncoded FPS 15fps 이상 유지. outbound-rtp.totalEncodeTime 증가량 / framesEncoded 증가량 기준선 대비 50ms 이내. qualityLimitationReason이 "cpu"가 아님. 지원자 화면에서 모든 알림 마스킹 정상 확인.

2.2 테스트 환경

클라이언트 컴퓨터 사양 요구사항

- CPU: Intel Core i5 (6세대 이상) 프로세서
- RAM: 8GB DDR4 이상

2.3 테스트 절차 요구사항

- 수집된 모든 getStats() 데이터 및 OS 모니터링 결과는 각 테스트 케이스 ID와 명확히 연결되어 정리되어야 한다.
- 측정된 KPI 값(계산된 비트 전송률, FPS, 인코딩 시간, CPU/메모리 사용량 등)은 예상 결과의 성능 목표치와 비교 분석되어 Pass/Fail 판정에 활용된다.

4. AI 도구 활용 정보

사용 도구	Gemini
사용 목적	테스트 계획 및 케이스 구조화, getStats() API 중심의 성능 측정 방법론 반영, 사용자 제약조건(도구 제한, 동시 테스트 제외, 특정 섹션부터 작성, "이전과 동일" 미사용 등)에 따른 반복적인 문서 수정 및 내용 상세화
프롬프트	"WebRTC 성능 측정 방법 문서를 참고해서 테스트 케이스를 제작해줘"
반영 위치	문서 전반 (사용자 요청에 따른 지속적인 내용 수정, 추가 및 상세화)
수작업 수정	구체적인 지침 및 프로젝트의 고유한 맥락에 맞춰 반복적으로 내용을 정제하고 보완함.