아티와 팬이 더 가까워지는 플랫폼,

fromm

QA Portfolio

서비스의 안정성과 사용자 경험을 지키는 QA

문제를 빠르게 발견하고, 더 큰 위험을 줄이기 위해
기획 · 개발 · 운영과 함께 좋은 배포를 만드는 과정을 담았습니다.



01 협업으로 만드는 품질

기획 · 개발 ·QA 가 함께 품질을 높일 수 있는 협업 환경 구축

02 작은 단위의 실험

품질 환경의 작은 변화를 시도하고 결과를 확인하며 개선

03 지속적인 문제 정의

변화를 관찰하고 다음 문제를 발견

04 리스크 사전 차단

기획 단계부터 위험을 발견하고 출시 전 감소

결함의 발견에서 개선 방식의 정착까지

서비스 구조와 검증 범위를 보여준 뒤,
실제 결함 · 원인 · 개선 적용 ·
성과와 한계를 순서대로 정리했습니다.

CONTENTS

문제 발견에서 기준 변화까지의 흐름

15페이지 구성 · 환경 → 결함 → 개선 적용 → 결과와 배움의 순서

01 **환경과 수행 범위** — 복합 팬덤 서비스 구조, 디바이스·도구·검증 범위 — 03–04

02 **결함과 원인 분석** — 실제 결함 사례, 비즈니스 영향, 팀 인터뷰 — 05–06

03 **Shift-left 적용과 공유 체계** — 참여 시점 변화, Jira·Slack·Playwright의 병행 도입 — 07–09

04 **문화 실험과 TC 기준** — 협업 문화 개선 실험, TC 선정 기준, 제품 협업 — 10–12

05 **성과, 한계, 원칙** — 개선 성과 종합, 남은 과제, 마무리 원칙 — 13–15

경력 및 QA 수행환경
복합 팬덤 서비스의 검증 범위를 소개합니다.

모바일·웹·백오피스·백엔드가 연결된 서비스에서 QA를 수행했습니다. 실제 검증 대상, 디바이스 환경과 도구를 함께 정리했습니다.

주요 경력

2022.10 – 2023.02
에스디지에이드
풀스택 개발 · QA
개발과 기능 검증을 함께 수행

2023.06 – 2024.08
노머스 · 운영기획팀 QA
신규 기능·회귀 테스트 / Jira 이슈 관리
기획·개발·디자인팀과 협업

2024.08 – 2025.08
노머스
물류 IT 기획 · 개발
업무 자동화와 운영 시스템 개선

fromm 한눈에 보기

아티스트와 팬을 연결하는 복합 팬덤 서비스

메시지, 커뮤니티, 멤버십, 상품 구매 등 여러 접점이 연결되어 있어 기능 결함은 팬 경험뿐 아니라 아티스트·계약사 대응에도 영향을 줄 수 있습니다.

팬 이용 경험 | 모바일 ↔ 웹 | 권한/데이터 정합성 | 운영 안정성

복합 팬덤 서비스 구성

모바일 · 웹 · 백엔드·백오피스가 연결된 구조에서 QA를 수행했습니다.

모바일 APP

메시지
아티스트/팬 1:1 핵심 이용 흐름

웹뷰
앱 내부에서 열리는 주요 페이지 동작

팬카페
게시물 · 멤버십 · 권한 기반 기능

↔

웹 WEB

커머스
상품 탐색, 결제, 주문 상태 확인

모바일 구매
모바일 환경에서의 결제와 권한 반영

현장 구매
오프라인 현장과 연결되는 구매 흐름

SYSTEM 백오피스 · 백엔드 로직 · 웹/앱 데이터·권한 정합성 등

백오피스
구매관리 · 운영기능 · 파트너 서비스 등 전반

백엔드 로직
백엔드 업데이트 시 서비스 환경 전반 QA

웹/앱 정합성
플랫폼 간 동기화 · 구매 상태 · 이용 권한 확인

📱 모바일 디바이스 범위

삼성 · 애플 기본 기기 외에도 **Flip · Fold · 중국 제조사 기기** 등 다양한 단말 환경을 고려해 검증했습니다.

웹·브라우저 범위

MacBook / Mac / Windows 환경에서 **Safari · Edge · Chrome** 등을 확인하며 웹 구매 흐름과 화면 표시를 검증했습니다.

사용 도구

Slack | Notion | Jira | Playwright | Python

수행 업무 및 개선 성과
구조 · 자동화 · 문화에 접근했습니다.

TC 작성·자동화·프로세스 개선·플래닝을 직접 수행했습니다. 효율성만으로 해결되지 않는 것은 구조로, 구조만으로 풀리지 않는 문제는 문화적 접근으로 시도했습니다.

직접 수행한 QA 업무

01 · TC
TC 작성

기획서를 해석하고 정상·예외 분기, 권한, 환경 조합을 정리해 간이/상세/통합 TC로 확장했습니다.

간이 TC · 상세 TC · 통합 TC

02 · AUTO
자동화

Playwright로 반복 세팅과 기본 회귀 검증을 보조해, 수작업으로 흩어지던 QA 준비 시간을 줄였습니다.

Playwright · Python

03 · PROCESS
프로세스 개선

우선순위, 위험 요소, 원인 구분, 진행 상태를 구조화해 QA가 뒤늦게 붙는 흐름을 바꾸고자 했습니다.

Jira · Slack · 문서 기준

04 · PLAN
플래닝

배포 일정과 리소스를 고려해 검증 범위와 담당을 배분하고, Hot-fix까지 포함한 일정을 사전 브리핑했습니다.

우선순위 · 일정 · 리소스

AS-IS → TO-BE · 세 가지 개선 축

01 · 구조 개선

상황·문제에 맞춘 QA 체계로 변경

기초적인 QA 프로세스를 명료화한 뒤,
상황과 문제에 맞춰 더 효율적이고 안전한 검증 체계로 변경

핵심 요소 사전 합의 | 우선순위 전달 | 리스크 선확인

02 · 자동화 및 리포팅

반복 QA 자동화 · 일정과 리포트 공유

```
01 / PLAYWRIGHT
✓ 반복 작업 및 QA 세팅 자동화
✓ 핵심 기본 회귀 확인 지원

02 / REPORTING
✓ QA 일정 사전 공유
✓ Jira 이슈 분류 + Slack 일일 리포팅
```

03 · 문화 개선

소통 오류를 협업 환경에서 다시 접근



책모임 시도
대화를 나눌 새로운 접점 탐색



축구 모임
업무 밖 관계 형성과 의견 교류



한강 피크닉
가벼운 만남을 통한 심리적 거리 축소

관찰된 결과 · 정량 / 정성

QA 리소스 확보
2배 이상
동일 스펙 가정 기준

QA 중 케이스 수
20~30% ↓
진행 중 등장 이슈 감소

배포 후 주요 이슈
0.5 → 0.2~0.3
월평균 횟수

정성적 반응
기획 우선순위와 일정 확인이 쉬움
개발 위험 요소와 FIX 타이밍 조율에 도움
문화 개선 소통 접점 확대를 위한 실험 시작

프로세스의 변화는 왜 필요했을까?

복합 팬덤 서비스에서는 작은 기능 결함도 팬의 이용 경험과 아티스트·계약사의 신뢰에 연결될 수 있었습니다. **오류의 건수보다 그 오류가 막는 비즈니스 흐름**을 먼저 이해해야 했습니다.

왜 서비스 품질이 중요한 비즈니스였나?

01 / 이용 경험
아티스트와 팬의 핵심 접점

메시지·팬카페·멤버십처럼 사용자가 반복해서 방문하는 기능을 운영했습니다.

02 / 수익과 권한
구매 이후 곧바로 이어지는 이용

상품 구매와 결제, 멤버십·채팅 이용 권한이 웹과 앱에 걸쳐 연결됐습니다.

03 / 계약사 관계
아티스트·엔터테인먼트 대상 서비스

콘텐츠·커머스 운영을 지원하는 사업 구조여서 장애가 계약사 대응으로 확대될 수 있었습니다.

작은 결함이 커지는 경로

웹뷰·구매·권한 오류
화면 미노출, 이용 권한 누락, 기능 동작 불일치

→

팬의 핵심 이용 중단
아티스트 게시물 확인 실패, 채팅 접근 차질 등

→

불만·긴급 대응 확대
아티스트 컴플레인과 CS·Hot-fix 부담

→

사업상 위험
유사 장애가 반복되면 서비스 신뢰 저하와 계약사 대응 위험으로 확대

실제 사례: 아티스트 화면 미노출로 직접 컴플레인이 발생했으며, 유사한 장애가 Hot-fix 및 비즈니스 위기로 이어진 경험이 있었습니다.

탐색적 QA 발견
아티스트 웹뷰 미노출
아티스트/관리자 화면에서 주요 게시물 페이지가 보이지 않음

탐색적 QA 발견
웹 구매 후 앱 미반영
구매·권한 부여 이후 모바일 기능 이용 차질

TC·리소스 한계
멤버십 게시물 권한
재검수 여력 부족으로 권한 조건 점검에 공백

기획·QA 동작 차이
예상 동작 불일치
기획자의 의도와 QA 판단 기준을 확인하며 발견

PROBLEM DEFINITION

결함 발생 뒤 수습하는 QA만으로는 부족했습니다.
중요한 이용 흐름과 위험 요소를 더 일찍 확인할 구조가 필요했습니다.

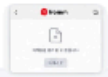
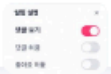
※ 위 네 결함의 영향과 별개로, 유사 장애에서 발생한 Hot-fix·비즈니스 위기 사례를 구분하여 서술했습니다. 계약 파기 등이 실제로 발생했다고 주장하지 않습니다.

결함 분석에 협업팀의 목소리를 더해, QA 운영의 개선 방향을 찾았습니다.

서로 다른 네 결함의 **검증상 한계**를 구분하고,
개발·기획·QA의 의견을 직접 확인했습니다.
기술적 결함 원인과 운영상 개선 과제는 구별했습니다.

01 결함에서 확인한 검증상의 한계

5페이지의 동일 사례 4건 · 기술적 원인 단정 아님

	주요 결함	발견 방식	QA 관점의 한계 / 질문	개선 검토 방향
01	 **아티스트 게시물 웹뷰 미노출** 아티스트·관리자 화면에 게시물 미표시	탐색적 QA	핵심 동작을 개발 전부터 중요 검증 대상으로 합의했다면?	핵심 기능·위험 요소를 사전에 합의
02	 **웹 구매 후 모바일 기능 미반영** 구매·권한 부여 후 모바일 기능 미작동	탐색적 QA	웹에서 완료된 처리가 앱 기능에도 반영되는지 확인 필요	웹 → 앱 사용자 흐름까지 함께 점검
03	 **멤버십 게시물 권한 오류** 권한별 게시물 노출 오류	TC 작성 · 재검수 한계	여러 권한·상태의 조합을 충분히 재검수할 리소스 부족	QA 준비 및 회귀 검수 시간 확보
04	 **예상 동작 불일치** 기획자와 QA가 서로 다른 동작 예상	기획자와 공동 확인	동일 기능의 기대 동작과 중요도에 대한 이해가 달랐음	기획·개발·QA가 동작 기준을 조기 합의

02 기획·개발·QA 인터뷰에서 확인한 요청

실제 의견을 요약한 문장 · 직접 인용 아님

⌘ 개발팀 — 우선순위 · 위험 공유

- 무엇을 우선해야 할지 명확하지 않음
- 어떤 부분에 위험이 있는지 파악이 어려움
- QA에 확인해야 할 사항을 미리 공유하고 싶음

▤ 기획팀 — 일정 · QA 의견

- QA 일정이 어떻게 진행되는지 파악이 어려움
- 기획의 우선순위를 QA에 전달할 방법이 모호함
- 기획 단계부터 QA의 의견을 듣고 싶음

✓ QA팀 — 사전 예방 · 정보 확보

- 사전에 막을 수 있는 위험이 있다고 판단
- 기획의 우선순위를 명시적 기준보다 감각적으로 파악

종합 문제정의

각 팀에는 필요한 정보와 의견이 있었지만,
이를 사전에 연결해 검증 기준으로 만드는 과정이 부족했습니다.

다음 페이지 → Shift-left 선택
사전 협의 · 간이 TC · QA 참여 시점 앞당기기
결함의 기술적 원인을 모두 소통 문제로 환원하지 않음

QA 참여 시점을 앞당겨,
핵심 위험을 먼저 합의하는 프로세스로 바꿨습니다.

기획 확정 후 뒤늦게 검수하던 방식에서 벗어나, **기획·개발·QA의 사전 논의와 간이 TC**를 통해 중요한 부분을 먼저 검토했습니다.

TO-BE · 실제 변경한 QA 프로세스 10단계

흰색 = 유지된 단계 / **핑크 = 새로 추가하거나 시점·방식을 변경한 단계** · 좌→우, 위→아래 순서

01 기획서 초안 작성 기획자 요구사항 초안 정리	02 기획·개발 논의 기능 구현 방향 및 기술적 조건 협의	
03 기획·개발·QA 3자 논의 (추가) 핵심 동작·위험 요소·검증 포인트 사전 정렬	04 기획서 최종 작성 + QA 기획 착수 (앞당김) 기획 확정 전부터 검증 범위 검토 시작	
05 간이 TC · 테스트 세팅 준비 (변경) 주요 분기·우선순위를 먼저 정리해 변경에 대비	06 QA 기획 브리핑 (추가) QA·기획·개발이 위험·범위·일정 공유	
07 세부 TC 작성 · 일정 산정 (조정) 일반: TC 상세화 / 긴급: 일정 산정 우선	08 QA 실행 합의된 중요도 기준으로 테스트 수행	
09 배포 출시 전 검증과 이슈 처리 상태 확인	10 배포 후 이슈 확인 발생 결함 추적 · 다음 개선 항목 기록	

AS-IS 기획 확정 → 개발과 QA 준비 병행 → QA 수행 → 배포 → 이슈 확인 · 버그 FIX
TO-BE ③ 사전 3자 논의 · ④ 조기 참여 · ⑤ 간이 TC · ⑥ QA 브리핑 · ⑦ 세부화/일정 판단 추가

왜 Shift-left를 선택했는가?

기존 결함 분석과 이해관계자 인터뷰에서 발견한 요구를 각각 다른 방식으로 해결

01 핵심 요소를 미리 맞추기 위해

탐색적 QA에서 발견하던 중요한 기능 조건을 기획·개발·QA가 먼저 공유해, **우선 검증 대상과 위험 요소**를 사전에 인지하도록 했습니다.

02 기획 변경에 유연하게 대응하기 위해

완성된 상세 TC를 일찍 고정하기보다 **간이 TC로 주요 분기**만 먼저 정리해 변경 시 재작성 부담을 줄이고자 했습니다.

03 QA 준비 여력을 확보하기 위해

QA 참여를 기획 확정 이전으로 옮겨 기획서 검토, 테스트 세팅, 일정 산정을 분산하고 **개발·기획 의견을 조기에 반영**했습니다.

DECISION PRINCIPLE

"테스트를 더 많이"에서
"중요한 것을 더 일찍 합의하고 준비"로.

Shift-left 적용과 함께,
이슈 분류 기준을 만들고 품질 정보를 공유했습니다.

Shift-left 프로세스 변경과 병행해 Jira에 우선순위·원인 라벨을 추가했습니다. 이를 Slack 일일 보고로 연결하고, Playwright로 반복 검증 부담을 줄였습니다.

Jira

신규 라벨 도입

이슈 분류 기준 구축

기존 이슈 등록 → 중요도·원인별 데이터 축적

이전에는 없었던 **우선순위**와 **이슈 원인** 라벨을 추가했습니다. 추적과 공유에 앞서 **무엇이 중요한지, 왜 발생했는지** 구분할 기준부터 정리했습니다.

새로 추가한 라벨 — 핵심 변경

우선순위	기준 없음	HIGH / MEDIUM / LOW
이슈 원인	분류 없음	커뮤니케이션 / API / 디자인 / 앱 구현

현황 집계를 위한 기록 항목

플랫폼	Android / iOS / WEB / BE
처리 상태	FIX / NON-FIX / 논의 중 / 추후 개선

이슈별 중요도·발생 원인을 구조화해, 단순 티켓 건수뿐 아니라 반복되는 위험

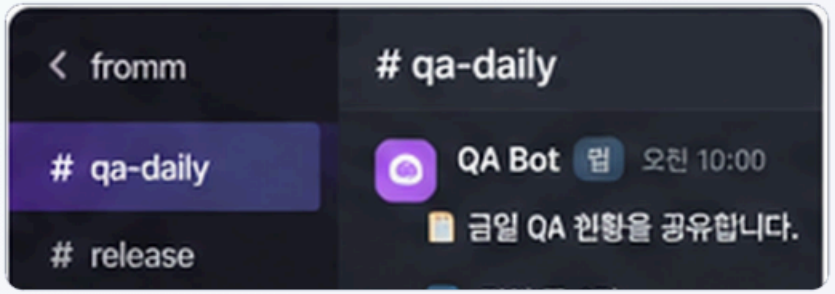
이슈 등록·라벨링 → 분류 데이터 축적 → Slack 일일 공유

Slack Daily

매일 공유

품질 현황을 팀 전체에 공유

플랫폼·처리 상태·남은 위험을 시각화



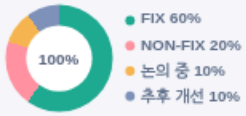
※ 실제 Slack 캡처가 아닌 디자인 재현 이미지

매일 작성한 QA 현황 보고 — 비율은 예시값

플랫폼별 이슈 비중



처리 상태별 이슈 비중



📌 잔여 이슈 수 및 중요 이슈 목록을 직접 작성

- 상 웹 렌더링 오류로 상세 페이지 미노출
- 중 상세 페이지 → 구매 페이지 이동 불가
- 최상 특정 상품 구매 불가

Playwright

반복 검증 보조

테스트 세팅·기본 회귀

상세 검증 전체를 대체한 것은 아님

test.spec.ts | login.spec.ts | store.spec.ts

```
1  import { test, expect } from '@playwright/test';
2
   test('아티스트 게시물 노출', async ({ page }) => {
3    await page.goto('/artist/123');
4    await expect(page.locator('.post-list')).toBeVisi
   });
```

※ 실제 코드 캡처가 아닌 디자인 재현 이미지

사용 목적과 범위

01 반복되는 테스트 환경 세팅 자동화
02 주요 기능의 기본 회귀 검증 지원
03 확보한 여력을 핵심·탐색적 QA에 배분

효과를 특정 절감 수치로 단정하지 않고, 반복 준비와 기본 확인에 활용한 범위를 설명합니다.

종합 인사이트

Jira에서 분류 → Slack에서 공유 → 축적된 데이터를 다음 개선 판단에 활용

Playwright는 별도로 반복 검증의 부담을 완화했습니다. 이 과정에서 이후 유형별 감소 폭을 확인할 수 있는 기록·공유 기반이 마련됐습니다.

※ 화면은 업무 방식 재현용 시안 · 파이차트 비율은 설명용 예시

품질을 꾸준히 공유하면서,
개선 효과와 남은 소통 문제를 함께 발견했습니다.

Shift-left와 Jira·Slack·Playwright를 병행한 뒤,
시간이 지나며 감소한 이슈와 여전히 남는 위험을 구분했습니다.

시간이 지나며 확인된 변화

QA 가용 리소스 확대
2배 이상

기획 초기에 검증 대상과 준비를 분산하면서 핵심 검증에 쓸 수 있는 여력이 증가

유사한 기획 스펙의 Jira 티켓
20~30% ↓

기획·개발·QA의 사전 논의 및 위험 재확인 이후 감소한 것으로 관찰

앱·웹·백엔드 관련 결함
각 20~30% ↓

사전에 위험한 영역을 점검해달라고 요청한 방식도 도움이 되었을 것으로 판단

※ 비슷한 규모의 기획을 비교한 운영 관측치이며 통제된 실험 결과는 아님

유독 남아 있던 치명적 이슈

치명적 이슈 중 커뮤니케이션 관련 이슈 비중

40~50%+

커뮤니케이션 오류 건수도 감소했지만,
치명적 이슈 중 차지하는 비중은 여전히 높게 관찰됨

새로 정의한 문제

"검증 절차와 도구를 개선한 것과 별개로,
사람들이 더 자연스럽게 의견을 나눌 협업 경험이 필요하다."

팀별 정성적 피드백 · 사전 협의와 공유에서 체감한 변화

기획자
"우선순위를 전하기 쉽고, QA 의견과 일정을 미리 확인할 수 있다."

개발자
"위험 요소를 먼저 듣고, QA 일정에 맞춰 FIX를 조율할 수 있다."

디자이너
"QA 일정과 리소스가 확보되어, 디자인 업무에 더 집중할 수 있었다."

운영팀
"QA가 독립적으로 일할 줄 알았는데, 안정감이 생기고 의견도 전달할 수 있어 좋았다."

다음 액션 **커뮤니케이션 오류의 비중을 보고, 프로세스 밖의 협업 경험도 실험하기 시작했습니다.**

다른 이슈보다 덜 줄어든 소통 오류,
이번에는 협업 경험 자체를 바꿔보았습니다.

Shift-left와 이슈 추적 체계를 정비하며 커뮤니케이션 오류도 약 20% 감소했습니다. 다만 치명적 이슈에서 차지하는 비중은 여전히 높아, 서로 의견을 편하게 주고받는 방법도 실험했습니다.

이슈 기록에서 확인한 출발점

**APP · WEB · BE 이슈는 감소했지만,
소통 오류는 같은 폭으로 줄지 않았습니다.**

Jira의 이슈 원인 라벨을 통해 구분한 당시 관찰입니다.

유형별 변화 요약	유사 기획 규모 기준
APP / WEB / BE	20~30% 감소
커뮤니케이션	감소 폭 제한

**새로운 질문: 업무 회의를 늘리는 것 외에,
서로 의견을 더 쉽게 나누는 방법은 없을까?**

순차적으로 시도한 팀 교류 방식

업무 외 상호작용 · 참여 방식 탐색

과거 인문학 커뮤니티 운영 경험도 참고해, 구성원이 자연스럽게 교류할 수 있는 형태를 탐색했습니다.

01 책모임



책을 매개로 업무 밖에서도 대화할 수 있는 기회를 마련했습니다.

관찰: 참여도가 낮아 지속적인 교류로 이어지기 어려웠음

02 축구 모임



공통의 신체 활동 속에서 역할을 맞추고 의견을 주고받을 기회를 만들었습니다.

변경: 말 중심 모임에서 공동 활동 중심으로 전환

03 한강 피크닉



업무 밖의 편안한 환경에서 부담 없이 이야기를 나누는 접점을 시도했습니다.

변경: 사적인 대화와 상호 이해의 접점 확대 시도

업무 현장에서 정성적으로 관찰한 변화

↗ **회의 내 대화량 증가**
회의에서 서로 의견을 제시하고 질문하는 장면이 늘었다고 관찰

↗ **업무 관련 대화 증가**
일상적인 업무 소통이 이전보다 활발해졌다는 현장 관찰

♡ **협업 만족도 개선**
팀 내 대화 분위기와 협업 경험에 대한 정성적 평가

성과 해석에서 구별한 것

✓ **확인한 변화:** 회의·업무 대화의 증가와 협업 분위기의 변화

! **확정하지 않은 것:** 교류 활동만으로 결함 감소가 발생했다는 인과관계

↳ **남은 한계:** 긴급 일정처럼 대화 여유가 없는 상황에서는 활용이 어려움

종합 인사이트

품질 개선을 위한 협업은 프로세스와 데이터뿐 아니라, 의견을 나눌 수 있는 관계와 환경도 함께 고려해야 했습니다.

※ 활동 이미지는 이해를 돕기 위한 재구성 · 관찰 결과는 정성적 회고이며 활동과의 인과관계는 검증되지 않았습니다.

요구사항과 위험도에 맞춰,
TC의 깊이와 검증 방식을 다르게 설계했습니다.

짧은 배포 주기와 기획 변경에 대응하기 위해
간이 TC로 먼저 범위를 합의하고, 기능·연동 위험에 따라
상세 검증과 탐색적 테스트를 조합했습니다.

검증 범위 판단 기준

①	**사용자·비즈니스 영향**	문제 발생 시 영향도
②	**변경 범위·복잡도**	신규 기능과 주요 분기
③	**연동·권한 위험**	웹·앱·백엔드 상태
④	**관계자 요청·리소스**	기획·개발·CS 및 일정

01 간이 TC

사전 협의와 검증 우선순위를 위한 설계 초안

언제 · 왜 사용했나?

기획이 바뀌기 쉬운 초기 단계에 주요 기능·분기를 먼저 정리. 기획·개발팀과 확인 범위, QA 준비, 소요 일정을 합의했습니다.

실제 문서 예시 · 멤버십 전용 게시물 간이 TC

사용자 상태 → 멤버십 유무 → 접근/반응

미가입: 전용 안내 | 가입: 게시물 열람 | 좋아요 노출 조건 | 댓글·알림

주요 권한 분기부터 합의 → 세부 케이스는 이후 확장

원본 간이 TC ↗

02 단위 TC (기능별 상세)

소규모 기능 배포의 필수 동작·예외 시나리오

언제 · 왜 사용했나?

회원가입 등 독립적인 기능 배포에서 필수 시나리오를 상세화. 동료가 동일하게 실행하거나 자동화에 활용할 수 있도록 작성했습니다.

실제 문서 예시 · 회원가입/이메일 인증 단위 TC

정상 동작	인증번호 입력 → 가입 흐름 진행
만료	인증번호 5분 경과 처리
재전송	10초 제한 · 이전 번호 재입력

입력값·경계 조건·예상 결과를 시나리오 단위로 기록

원본 상세 TC ↗

03 통합 TC

핵심 흐름과 플랫폼 간 연동을 확인하는 검증

언제 · 왜 사용했나?

안정적인 영역의 회귀 검증과 웹·앱·백엔드 연결 동작 확인. 구매·구독 이후 이용 권한과 사용자 상태를 함께 점검했습니다.

실제 문서 예시 · 구독/채팅 흐름 통합 TC

웹 구독 → 권한 반영 → 앱 채팅

구독 상태 변경 | 메시지 송수신 | 백그라운드 | 앱 재실행

개별 기능이 아니라 이어지는 사용자 경험을 함께 확인

TO-BE 프로세스에서 실제로 활용한 순서

기획·개발·QA 3자 사전 논의 › 간이 TC + 테스트 준비 › QA 브리핑 우선순위 합의 › 상세·통합 TC 및 일정 산정 › 검증 실행 + 탐색적 테스트

탐색적 테스트 병행

7 : 3
TC 기반 : 탐색적

경험적 운영 기준.
변경 폭·비즈니스 영향이 큰 경우 **6:4까지 조정**했습니다.

모든 기능을 똑같이 검수하지 않았습니다.

중요한 분기를 먼저 합의하고, 위험과 상황에 따라 검증의 깊이를 조절했습니다.

QA를 기획 단계로 앞당기자,
제품 동선과 개발 로직에도 의견을 낼 수 있었습니다.

검증 항목을 확인하는 데 그치지 않고,
기획자·개발자와 함께 사용자 흐름과 기능 조건을 검토했습니다.
실제 제안의 채택 여부와 결과는 구분해 기록했습니다.

사전 참여로 확대된 역할 | **QA의 질문과 의견**이 기획 우선순위와 구현 방식에 전달되는 구조를 만들었습니다.

01 사용자 동선 개선 제안

제품 기획 관점 · 관심 아티스트 탐색 흐름

당시 미채택 → 이후 관련 기능 반영

문제 제기	사용자가 관심 아티스트와 원하는 기능에 더 쉽게 도달할 수 있도록 **동선 변경을 제안**했습니다.
당시 결정	제안은 초기 논의에서 **채택되지 않았습니다.** 설득에 필요한 근거도 충분히 제시하지 못했습니다.
이후 변화	사용자 이탈 문제가 확인된 뒤, 관련 방향의 기능이 **나중에 도입**되는 것을 확인했습니다.

사용자 흐름 관점에서 제안

동선 문제를 설명하기 위한 개념도

검토한 관점

서비스 진입 → 관심 아티스트 발견 → 목표 기능 이용

당시에는 미반영 · 이후 관련 개선 확인

QA 관점 제시 → 우선순위 논의 → 후속 반영

※ 실제 화면·정확한 기능 흐름이 아닌 설명용 도식

↗ **배운 점** — 제품 개선 제안에는 QA의 문제 인식뿐 아니라 **사용자 데이터와 기획 우선순위를 설명할 근거**가 필요했습니다.

02 복잡한 기능 로직 단순화

개발 협업 관점 · 예외 조건과 예상 동작 검토

사전 논의에서 함께 개선

상황	복잡한 조건과 예상 동작을 검토하면서 **기능 로직을 함께 점검할 필요**가 있었습니다.
접근	QA만 판단하지 않고 **기획자·개발자와 동작과 조건을 함께 논의**했습니다.
결과	대화를 통해 복잡했던 조건을 정리하고, **로직을 더 단순하게 만드는 데 기여**했습니다.

복잡한 분기 → 합의된 동작 기준

구현 의사결정 과정을 설명하기 위한 개념도

논의 이전

조건 A ↘ 여러 분기 및 예외 ↗ 조건 B

공동 검토 이후

핵심 조건 확인 → 예상 동작 합의 → 단순화된 로직

※ 구체 구현 내용은 공개하지 않은 설명용 도식

⇄ **배운 점** — QA는 완성된 기능만 확인하는 사람이 아니라, **동작 기준을 합의하는 논의에 참여**할 수 있었습니다.

QA의 기여 방식

01 사용자 관점에서 문제를 제기

02 기획 우선순위와 함께 검토

03 개발 조건과 예상 동작을 합의

04 검증하기 쉬운 제품 구조에 기여

※ 두 사례는 서로 다른 경험입니다. 첫 번째 제안은 당시 미채택됐으며, 후속 적용과의 인과관계까지 입증한 것은 아닙니다.

프로세스를 바꾼 뒤의 변화를,
검증 여력·이슈 감소·배포 품질로 확인했습니다.

사전 협의, 간이 TC, 조기 QA 참여부터
Jira·Slack·Playwright까지 적용하며 관찰한 결과입니다.
비슷한 기획 규모를 비교한 당시 운영 기준으로 정리했습니다.

핵심 성과 요약

동일하거나 비슷한 기획 규모 비교 · 당시 운영 관측치

QA에 활용 가능한 리소스
사전 검토·세팅·우선순위 합의로 확보

1.0× → **2.0× 이상**

이전 — 상대 기준
이후 — 2배 이상

QA 준비 부담을 앞당겨 분산하고, 중요한 검증에 집중할 여력 확대

QA 수행 중 발생한 이슈
유사한 기획 스펙에서 Jira 이슈 비교

100% → **70~80%**

이전 — 기준 100
이후 — 70~80

약 20~30% 감소 · 초기 협의로 일부 위험을 사전에 확인

배포 후 주요 이슈 발생
월평균 주요 이슈 발생 빈도 비교

월 0.5회 → **0.2~0.3회**

이전 — 0.5회/월
이후 — 0.2~0.3회

출시 후 중대한 문제의 빈도가 감소한 것으로 관찰

플랫폼별 이슈 변화 · 무엇이 달랐는가?

APP	WEB	BE	별도 관찰
앱 구현 관련	**웹 관련**	**백엔드 관련**	**소통 관련 오류**
−20~30%	**−20~30%**	**−20~30%**	**약 20% 감소**
이전 유사 업무 대비 감소 관찰	검증 범위 조기 확인과 관련해 감소 관찰	사전 위험 재확인과 관련해 감소 관찰	이슈 건수는 감소했으나 치명적 이슈 비중은 여전히 높았음

수치 뒤에서 확인된 운영 변화

준비 방식
급박한 준비 → 사전 분산
기획 단계부터 QA 내용을 읽고 검토.
변경 시 간이 TC를 조정해 재작성 부담을 줄였습니다.

협업 방식
개별 판단 → 공동 검증 기준
기획자는 우선순위·일정을 공유하고,
개발자는 위험 요소와 FIX 시점을 사전 조율했습니다.

인과관계 해석은 구분: 사전 협의·테스트 방식 변경·도구 적용이 함께 진행됐으므로, 특정 활동 하나의 효과로 모든 감소 수치를 귀속하지 않았습니다.

종합 성과

검증의 시점을 앞당기고 판단을 공유한 결과,
QA 여력과 출시 품질을 함께 개선했습니다.

비교 기준: 당시 비슷한 기획 규모·업무 조건의 내부 관측
정밀한 대조실험이나 단일 활동별 효과 측정은 수행하지 않음

프로세스는 개선됐지만,
모든 상황에서 같은 방식이 통하지는 않았습니다.

QA 리소스와 출시 품질의 개선을 관찰했습니다.
동시에 긴급 변경, 검증 범위, 협업 방식과 성과 측정에서
남은 한계와 추가로 고민할 문제도 확인했습니다.

실제 운영에서 확인한 4가지 한계

긴급 일정에서 협의가 어려움

계약사 요청으로 기능·일정이 급변하면 기획 초기부터 QA가 참여할 시간이 충분하지 않았습니다.

한계 Shift-left 효과를 모든 배포에 동일하게 적용할 수 없음

검증 범위 선택이 필수

기능과 사용자 상태가 복잡할수록 모든 분기·예외를 같은 깊이로 재검수하기는 어려웠습니다.

판단 핵심 기능·권한·연동 위험을 먼저 합의하고 검증

소통 오류는 상대적으로 잔존

커뮤니케이션 관련 오류도 약 20% 감소했지만, 치명적 이슈에서의 비중은 여전히 높았습니다.

추가 실험 업무 외 교류도 시도했지만 직접적인 개선 효과는 분리 측정하지 못함

성과 관찰과 인과관계는 별개

비슷한 기획 규모의 Jira 기록과 운영 경험을 비교했지만, 개선 활동별 효과를 분리한 실험은 아니었습니다.

측정 한계 감소 수치를 특정 도구·회의·문화 활동의 단독 성과로 볼 수 없음

일정에 따라 달라지는 QA의 선택

프로세스는 고정된 절차가 아니라 판단의 기준이었습니다.

정기 배포 사전 준비가 가능한 상황

사전 3자 협의 → 간이 TC → 상세 TC → 검증

요구사항을 일찍 읽고 위험 영역과 우선순위를 함께 정해, QA 준비 부담을 분산했습니다.

긴급 변경 일정·기획이 갑자기 바뀐 상황

변경점 확인 → 간이 TC 조정 → 범위·일정 합의 → 중요 검증

상세 TC를 일괄 완성하기보다, 바뀐 기능과 남은 시간에 맞춰 검증 범위를 다시 결정했습니다.

완벽한 사전 차단은 어려웠습니다.

그러나 중요한 위험을 더 이른 단계에서 발견할 가능성을 높이고, **발견하지 못할 수 있는 영역까지 팀과 공유하는 것**이 QA의 역할이라고 판단했습니다.

핵심 인사이트

모든 문제를 없애는 프로세스보다,
상황에 따라 위험을 줄이는 판단이 중요했습니다.

다음 장에서는 이 경험을 통해 정리한
네 가지 QA 원칙을 소개합니다.

완벽한 배포는 불가능하지만,
더 나은 출시 방식은 만들 수 있습니다.

QA 프로세스를 직접 설계하고 개선하면서,
기능 검증을 넘어 **품질을 바라보는 기준과**
함께 일하는 방식에 대한 네 가지 원칙을 얻었습니다.

01 현실 인식



완벽한 배포는 불가능합니다.

모든 결함을 없앨 수는 없습니다. 그러나 중요한 위험을 출시 전에 발견하고 줄이려는 노력은 계속해야 합니다.

> 완벽함을 약속하기보다,
> 사전에 중요한 위험을 줄이기

02 사전 차단



위험은 가능한 한 앞단에서 차단합니다.

워터폴 방식에서는 기획·요구사항 단계에서 위험을 발견할수록 수정 리소스와 파급 영향을 줄일 수 있었습니다.

> 발견된 오류에 대응하는 데서
> 사전 예방으로 무게중심 이동

03 QA의 역할



QA는 배포를 막는 역할이 아닙니다.

검증 기준과 위험을 명확히 공유해 기획자·개발자가 더 원활하게 출시할 수 있도록 돕는 역할입니다.

> 품질을 지키면서,
> 출시가 매끄럽게 이어지도록 돕기

04 협업 환경



좋은 협업 환경이 품질을 높입니다.

서로의 우선순위와 우려를 충분히 나눌 수 있을 때, 기획·개발·QA 각자의 역량을 더 잘 발휘할 수 있습니다.

> 좋은 품질을 만드는 일은
> QA팀 혼자만의 몫이 아닙니다

MY QA PRINCIPLE

QA는 마지막에 결함을 찾아내는 관문을 넘어, 모두가 더 좋은 제품을 출시하도록 돕는 역할입니다.

이번 경험은 **사람·프로세스·검증 기준을 연결하는 일**이
안정적인 서비스 품질을 만드는 데
얼마나 중요한지 보여주었습니다.