IP01 · GIBALJA VISUAL MANUAL

통합 프로젝트 1 · 업무관리 웹 서비스 만들기

한 문장 목표: `회의 후속 조치 문제 → actor·화면·task state → UI·API·SQLite → session·권한·오류·충돌 → 307 tests·59 audit·6 regression → backup restore → Integrated Project Gate` 를 하나의 추적 가능한 서비스로 완성합니다.

LEARN BY SEEING · DOING · EXPLAINING

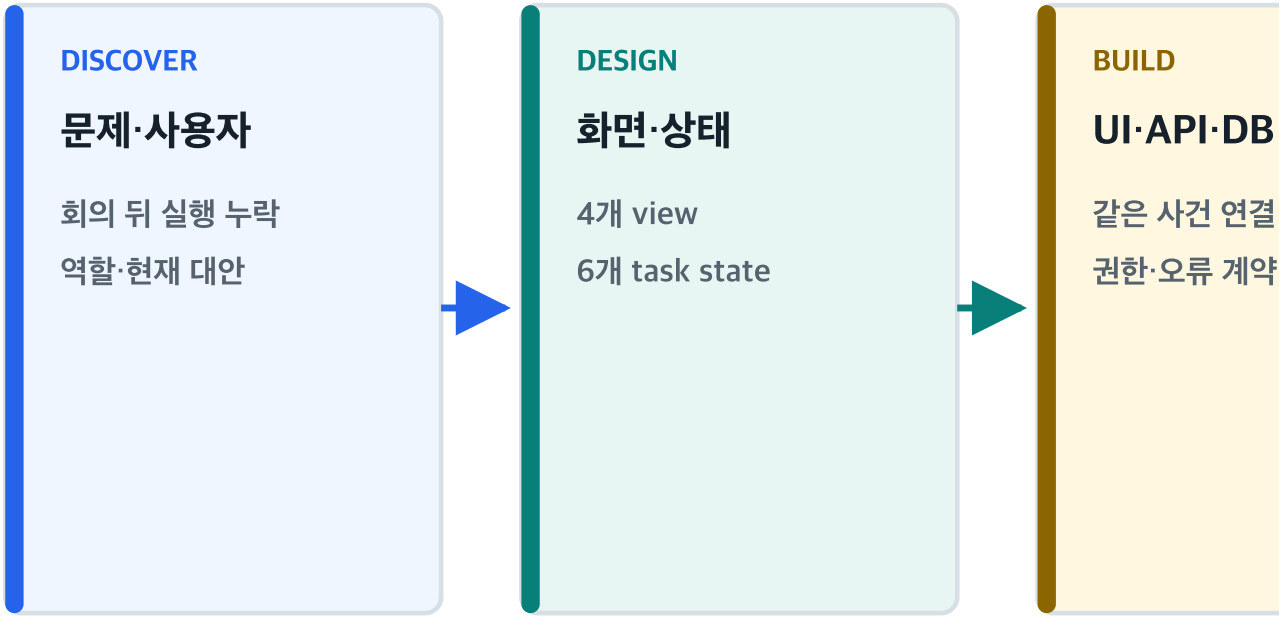
도표로 구조를 보고, 실제 화면에서 확인하고, 같은 사건을 직접 추적해 설명합니다.

학습 결과	확인 방법	완료 신호
사용자 흐름	Owner·Viewer·mobile 실제 화면	핵심 행동·읽기 전용·반응형 동작
서비스 계약	UI·API·DB same-event trace	task ID·version·event·audit 일치
안전·복구	권한·validation·충돌·restore	negative path와 recovery evidence
검증	12 controls·24 scenarios	24/24·critical12/12·orphan0
학습 package	도표15·화면9·워크북·용어300	PDF·web·preview

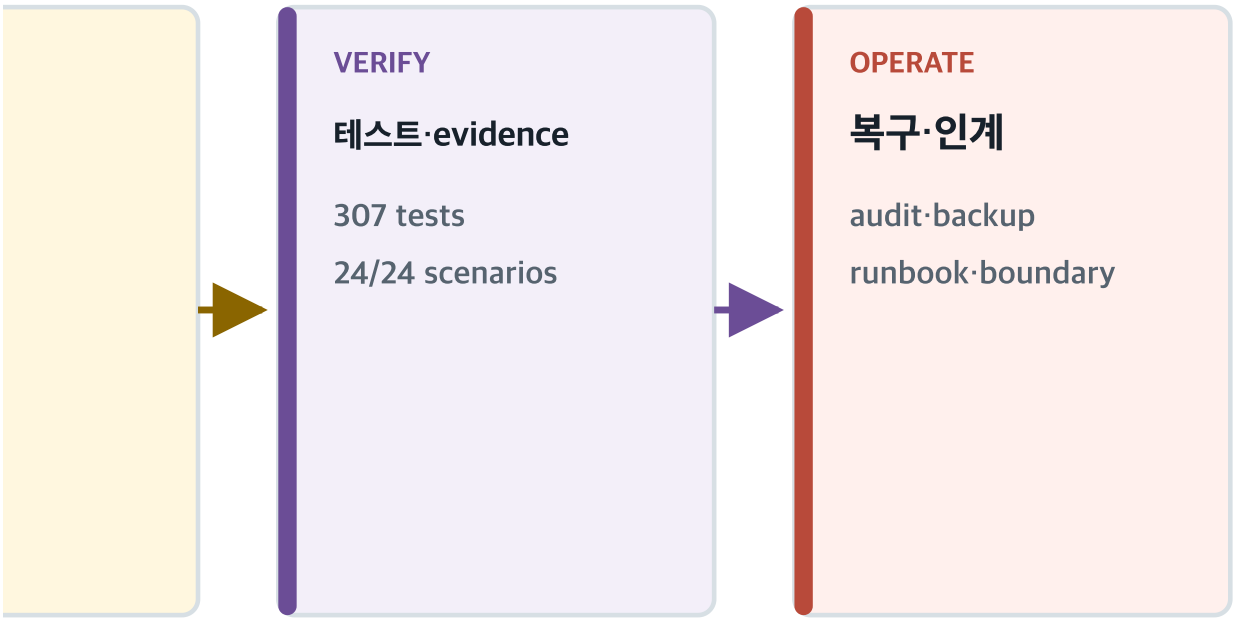
1. 문제에서 운영까지 evidence 여정 만들기

그림 1. 기획·구현·검증·운영을 하나의 사용자 사건으로 연결합니다.

문제에서 운영 인계까지 같은 업무 사건을 추적해 완성도를 판단합니다.



기획 → 구현 → 검증 → 운영 eviden



ce가 한 서비스 사건으로 이어집니다

핵심 원리
통합 프로젝트의 완성 기준은 화면 수가 아니라 사용자가 일을 끝내고 실패에서 회복하는 전체 흐름입니다.

업무관리 서비스는 목록을 보여 주는 화면만으로 완성되지 않습니다. 회의 뒤 실행 항목을 놓치지 않는다는 문제에서 시작해 actor·상태·화면을 설계하고, UI 요청이 API와 SQLite transaction으로 이어지며, 권한 거부·충돌·backup 복구까지 재현되어야 합니다. IP01은 이 여정을 12개 통제와 24개 시나리오로 확인합니다.

관점	IP01에서 보이는 것	확인 evidence
Discover	사용자·문제·현재 대안	PRD·scope·metric hypothesis
Design	역할·화면·상태	permission·view·state map
Build	UI·API·DB	same-event implementation
Verify	규칙·계약·브라우저	307 tests·59 audit·6 regression
Operate	audit·backup·인계	restore rehearsal·runbook

흔한 오해
보드 화면이 열리고 task card가 보이면 프로젝트가 끝났다고 판단합니다.

더 나은 판단
같은 task 사건이 화면·요청·권한·DB·audit·시험·복구에서 재현되는지 확인합니다.

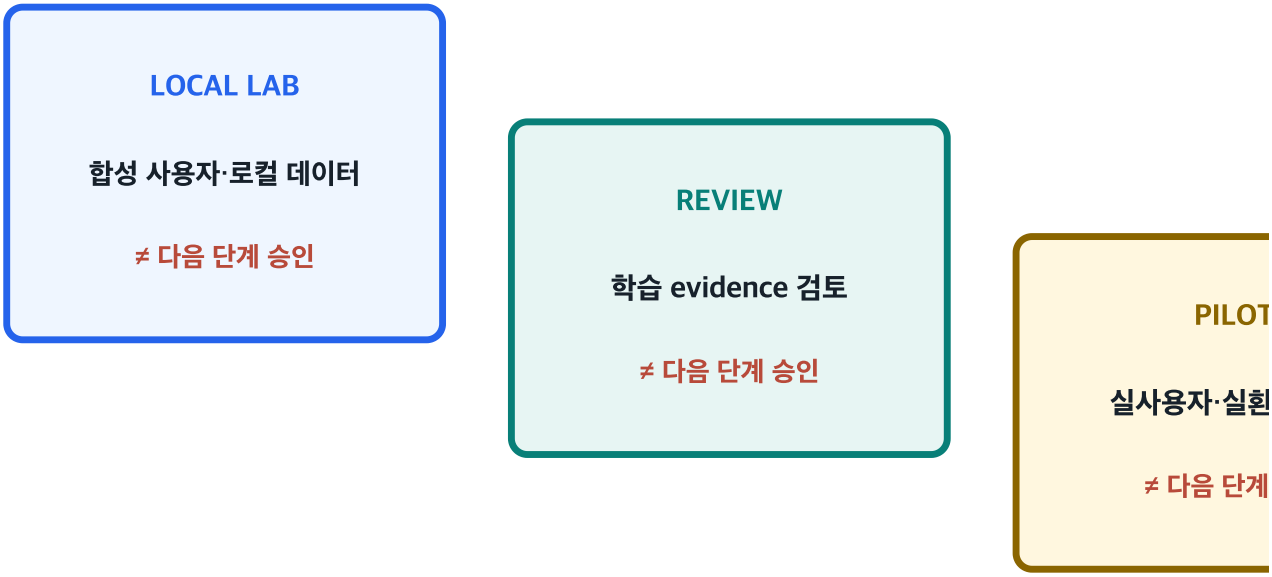
4단계 미니 실습

1. 회의 후속 조치라는 사용자 일을 한 문장으로 적습니다.
2. 일을 끝내기 위해 거쳐야 할 다섯 단계를 표시합니다.
3. 각 단계의 evidence 파일이나 화면을 연결합니다.
4. 한 단계가 실패했을 때 다음 복구 행동을 적습니다.

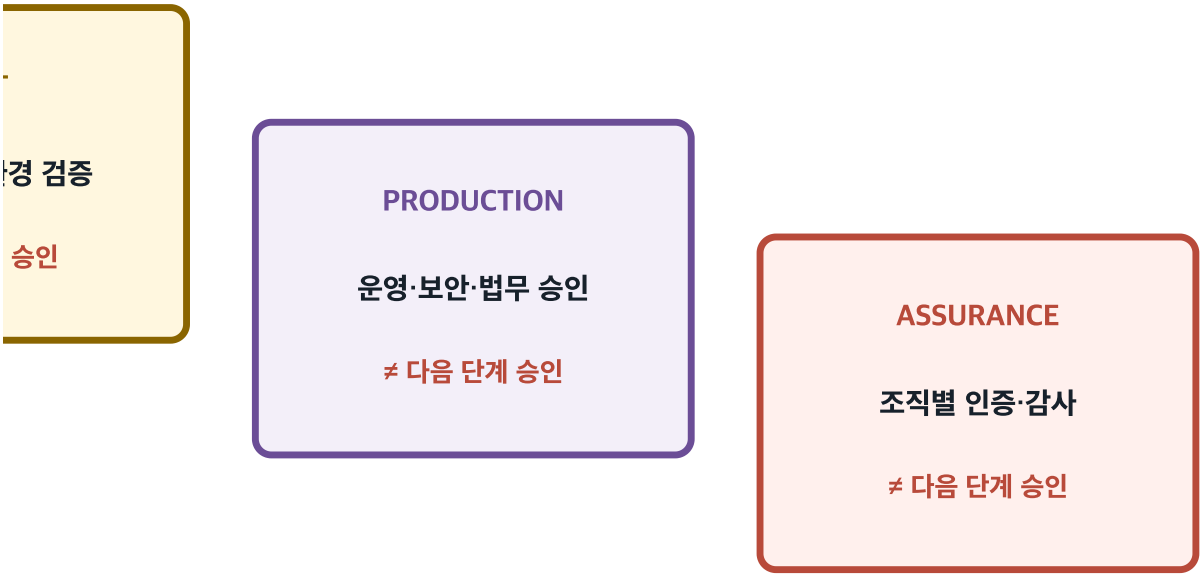
2. 학습용 서비스와 실제 운영 권한 구분하기

그림 2. 로컬 동작·학습 검토·실사용 pilot·production 승인은 서로 다른 단계입니다.

동작하는 데모도 실사용자 조사·운영 환경 검증·조직 승인을 자동으로 대신하지



| 않습니다.


경 검증
승인
PRODUCTION
운영·보안·법무 승인
≠ 다음 단계 승인
ASSURANCE
조직별 인증·감사
≠ 다음 단계 승인

핵심 원리
동작하는 로컬 서비스는 좋은 학습 evidence이지만 실제 로그인·개인정보·조직 보안·production 승인을 대신하지 않습니다.

IP01은 합성 actor와 합성 업무만 사용하고 Python 표준 라이브러리 서버를 로컬에서 실행합니다. session·CSRF·권한·validation 원리는 구현하지만 실제 identity provider, TLS 종단, 조직 비밀 관리, 개인정보 영향평가, 가용성 설계, 보안 인증은 범위 밖입니다. 완료 문구를 `integrated_project_review_ready`로 제한하는 이유입니다.

관점	IP01에서 보이는 것	확인 evidence
Local lab	합성 actor·SQLite	학습자
Review	24 시나리오 evidence	과정 reviewer
Pilot	실사용자·실데이터 경계	product·research owner
Production	배포·보안·법무	조직 authority
Assurance	감사·인증·위험 수용	qualified authority

흔한 오해
24/24 PASS를 production 보안 검증 완료로 표현합니다.

더 나은 판단
학습 Gate가 답하는 질문과 실제 운영 전에 남은 승인·환경·조사 질문을 함께 기록합니다.

4단계 미니 실습

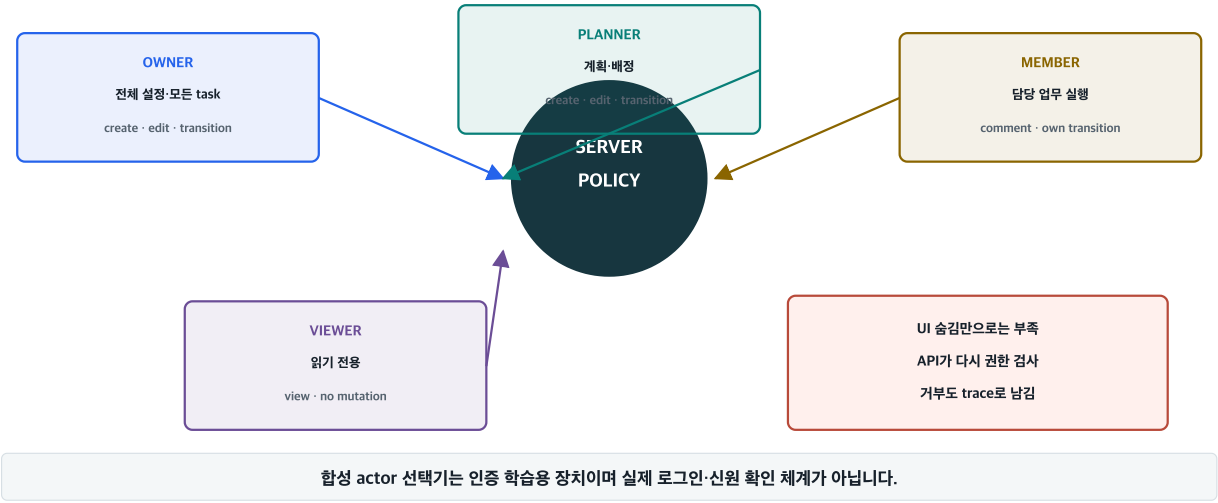
1. 실습이 사용하는 합성 자원을 적습니다.
2. 실제 환경에서 추가될 actor와 data를 적습니다.
3. 누가 어떤 승인을 해야 하는지 표시합니다.
4. 완료 문구에 authority boundary를 붙입니다.

3. 역할을 UI가 아니라 서버 정책으로 보호하기

IP01 · 03

역할별 화면 차이보다 중요한 것은 서버의 일관된 권한 판정입니다

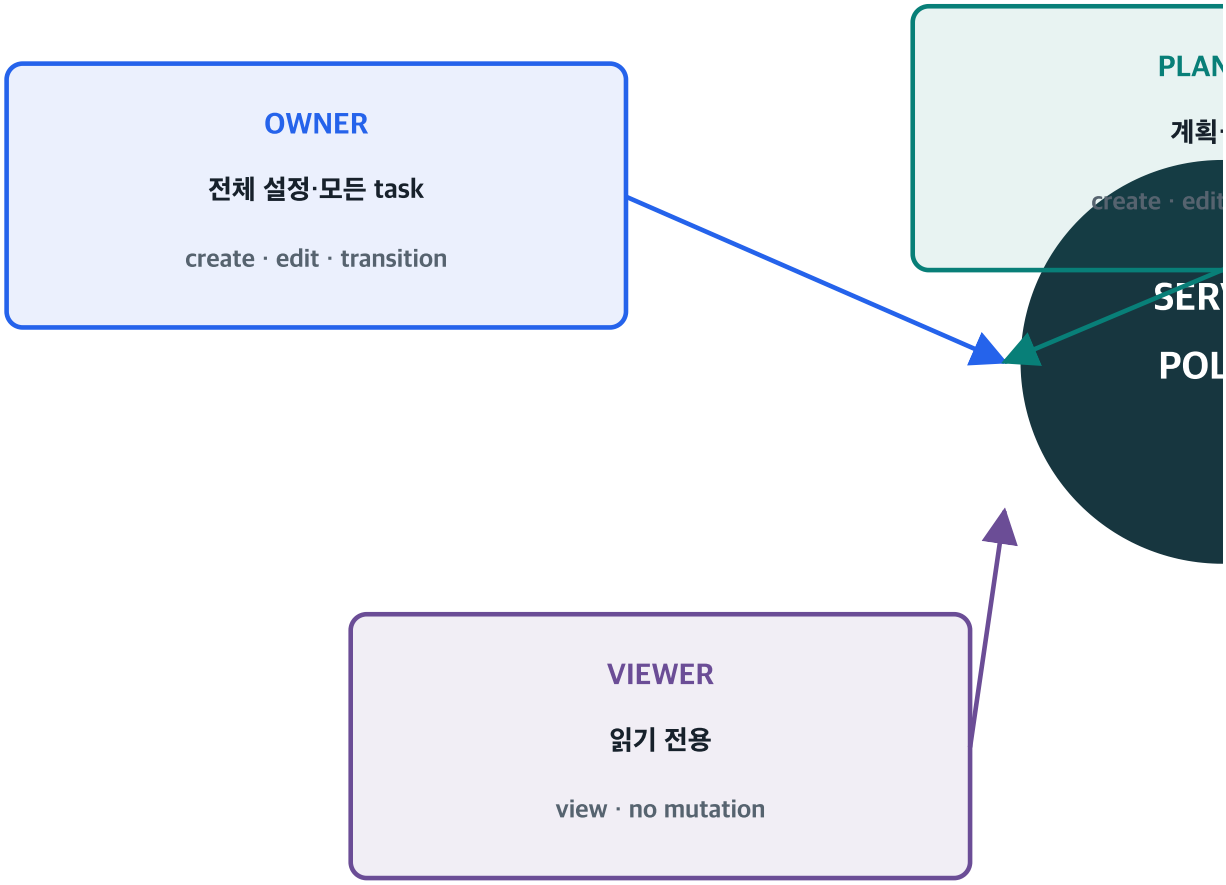
Owner·Planner·Member·Viewer가 같은 요청을 보내도 서버 정책에 따라 결과가 달라집니다.



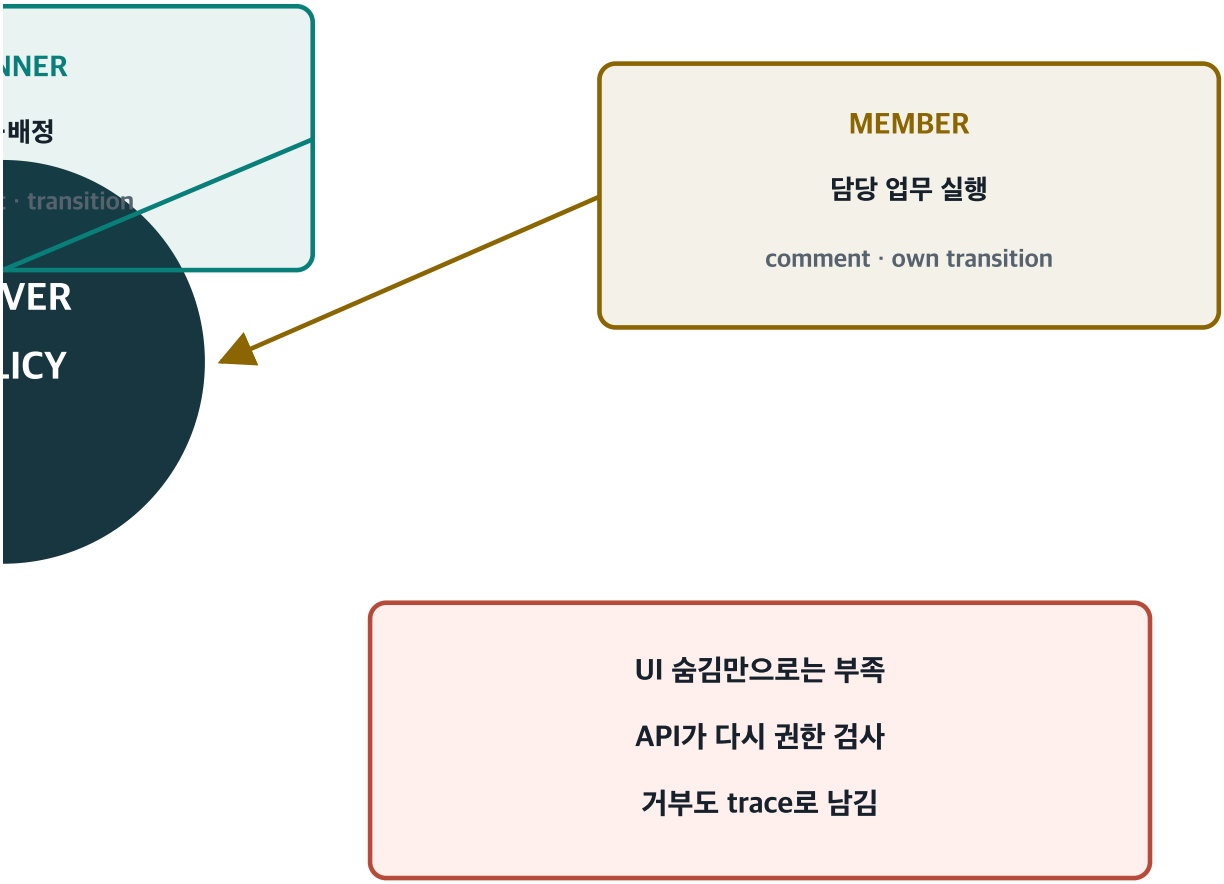
합성 actor 선택기는 인증 학습용 장치이며 실제 로그인·신원 확인 체계가 아닙니다.

그림 3. 네 역할의 행동을 서버가 resource와 함께 다시 판정합니다.

Owner·Planner·Member·Viewer가 같은 요청을 보내도 서버 정책에 따라 결고



가 달라집니다.


NNER
배정
· transition
VER
LICY
MEMBER
담당 업무 실행
comment · own transition
UI 숨김만으로는 부족
API가 다시 권한 검사
거부도 trace로 남김

핵심 원리
버튼을 숨기는 것은 사용성 처리이고, 권한을 지키는 곳은 서버입니다.

Owner와 Planner는 업무를 만들고 편집할 수 있고, Member는 담당 업무의 실행에 참여하며, Viewer는 조회만 합니다. 브라우저는 역할에 맞춰 명령을 감추지만 공격자는 직접 API를 호출할 수 있습니다. 그래서 서버가 session actor·role·target task·requested action을 다시 평가하고 거부 결과도 trace에 남깁니다.

관점	IP01에서 보이는 것	확인 evidence
Owner	전체 task·설정	create·edit·transition·comment
Planner	계획·배정	create·edit·transition·comment
Member	담당 업무 실행	view·comment·허용 transition
Viewer	읽기 전용	view only
Server	모든 mutation	session·role·resource policy

흔한 오해
Viewer 화면에서 새 업무 버튼만 숨기고 POST API는 그대로 허용합니다.

더 나은 판단
UI와 서버를 각각 시험해 Viewer는 버튼 0개이고 직접 mutation 요청도 거부되는지 확인합니다.

4단계 미니 실습

1. 네 역할의 허용 행동을 표로 적습니다.
2. UI에서 숨길 명령과 서버에서 거부할 명령을 나눕니다.
3. 다른 사람의 task라는 resource 조건을 추가합니다.
4. 허용·거부 양쪽 시험을 만듭니다.

4. 한 사용자 일을 수직 slice로 끝까지 완성하기

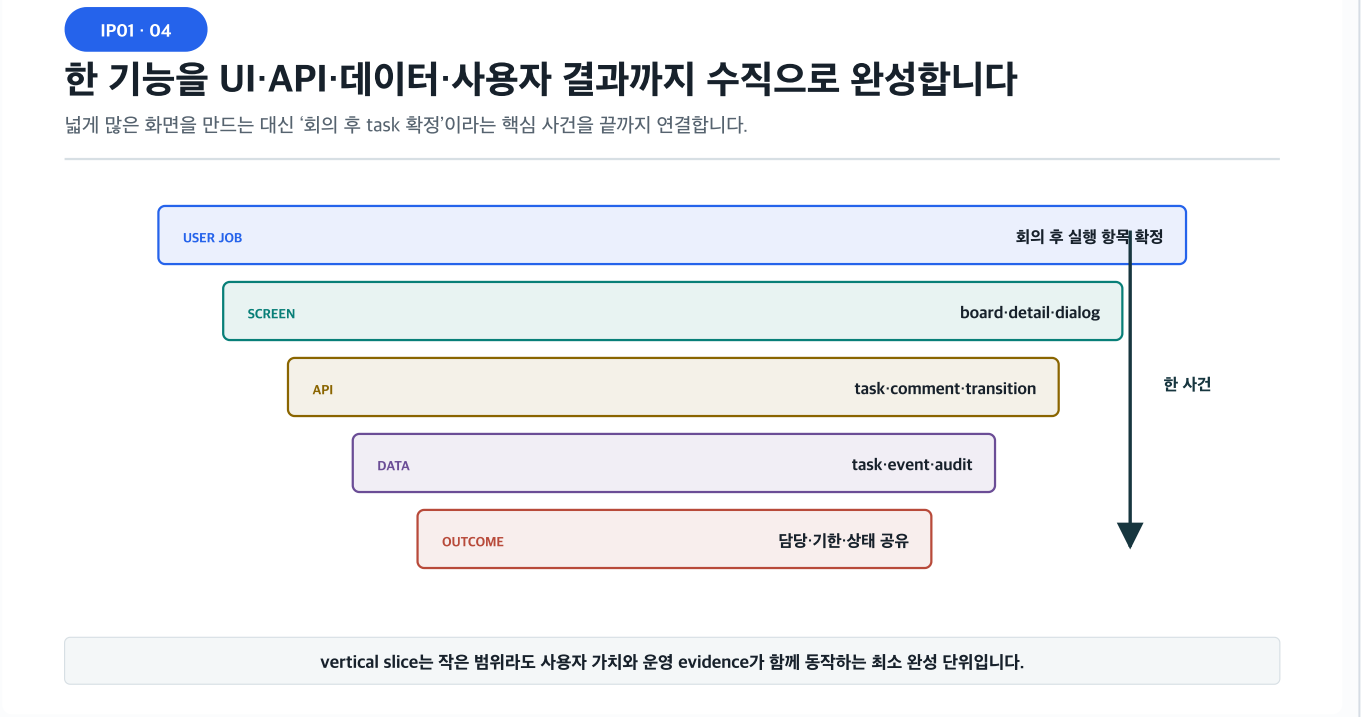
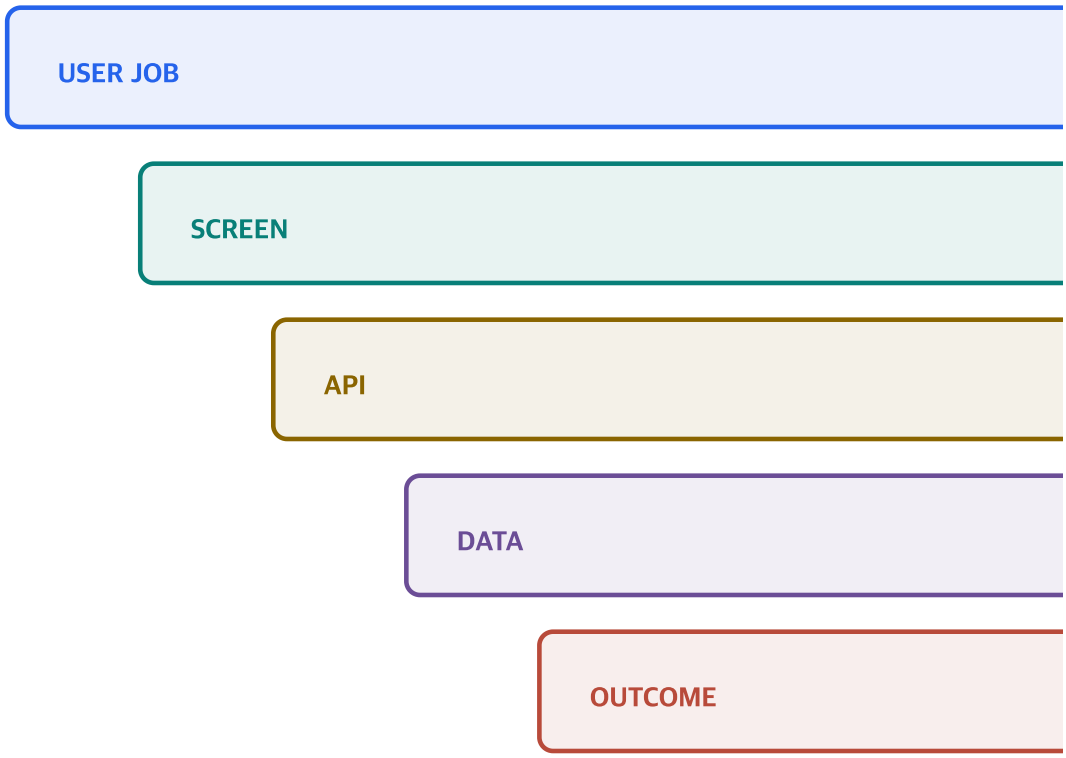
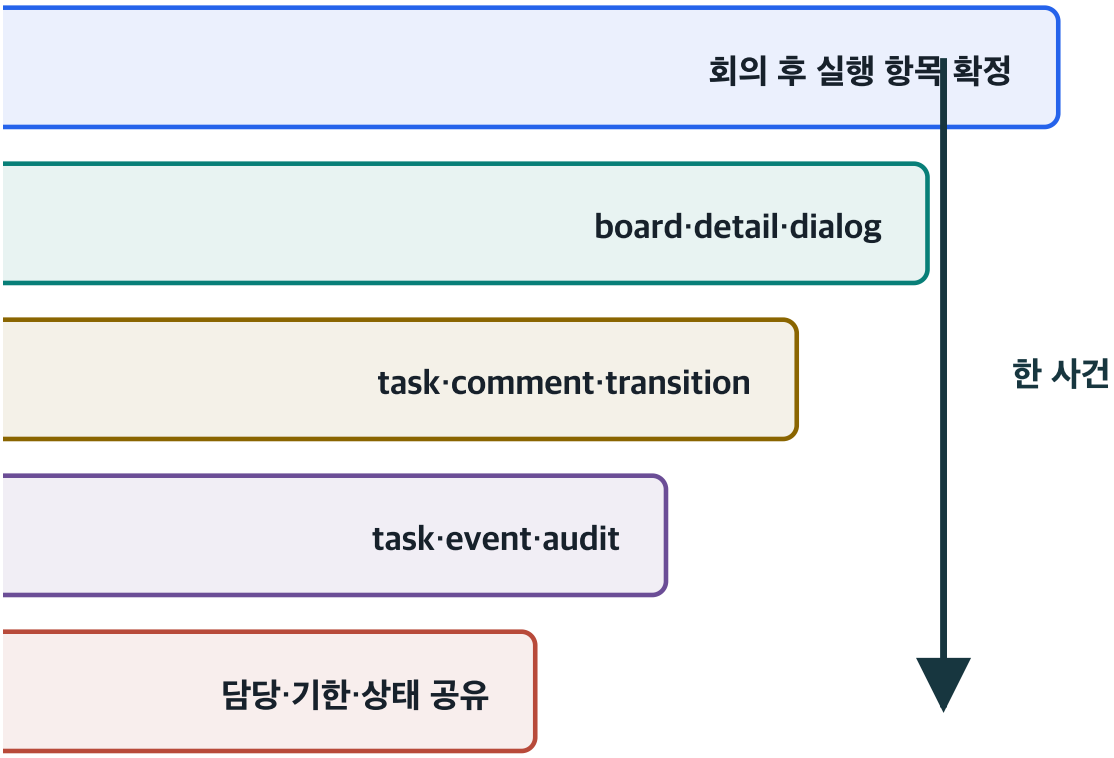


그림 4. 회의 후 task 확정이라는 한 일을 화면·API·데이터·결과까지 잇습니다.

넓게 많은 화면을 만드는 대신 ‘회의 후 task 확정’이라는 핵심 사건을 끝까지 연



결합니다.


회의 후 실행 항목 확정
board·detail·dialog
task·comment·transition
task·event·audit
담당·기한·상태 공유
한 사건

핵심 원리
넓고 얕은 기능 목록보다 작더라도 사용자 결과와 운영 evidence까지 연결된 slice가 더 강합니다.

IP01의 핵심 slice는 '회의에서 나온 실행 항목을 담당·기한·완료 기준과 함께 확정하고 상태를 추적한다'입니다. 이 일을 위해 board·detail·dialog가 있고 task·comment·transition API가 있으며 task·event·audit 데이터가 남습니다. 각 계층은 따로 존재하는 것이 아니라 같은 task ID와 version을 공유합니다.

관점	IP01에서 보이는 것	확인 evidence
User job	회의 후 실행 항목 확정	누락·재확인 감소 가설
Screen	board·detail·dialog	scan·decide·act
API	create·edit·comment·transition	status·problem contract
Data	task·comment·event·audit	identity·history·trace
Outcome	담당·기한·상태 공유	합성 metric hypothesis

흔한 오해
보드·달력·채팅·보고서 화면을 많이 만들지만 어떤 사용자 일이 끝나는지 설명하지 못합니다.

더 나은 판단
하나의 acceptance trace를 고르고 모든 계층의 식별자·입력·결과·실패를 연결합니다.

4단계 미니 실습

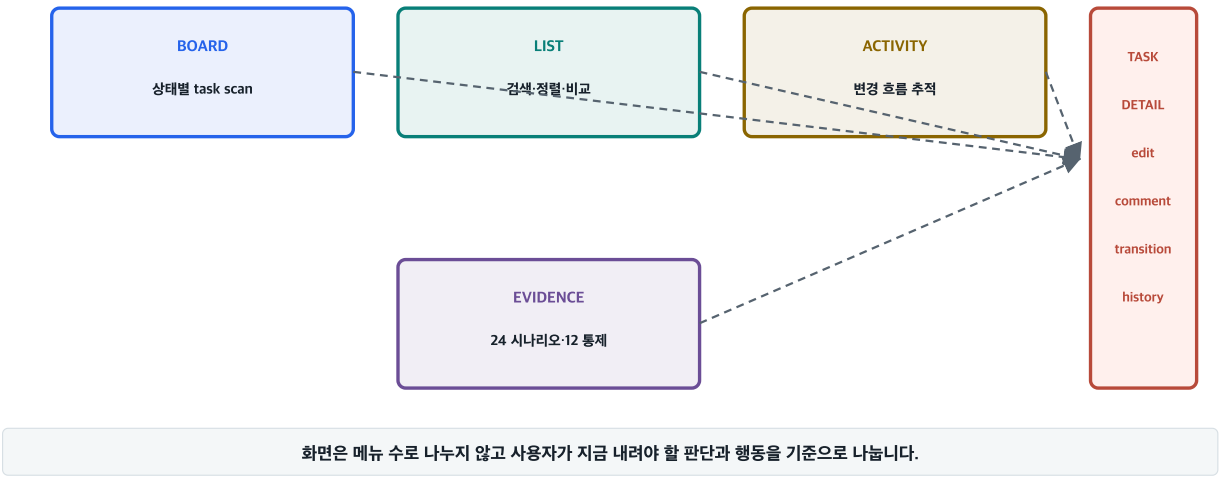
1. 핵심 user job을 동사로 씁니다.
2. 필요한 최소 화면과 API를 연결합니다.
3. 같은 사건을 저장할 데이터 행을 정합니다.
4. 사용자 결과와 guardrail을 적습니다.

5. 화면을 메뉴가 아니라 판단 질문으로 나누기

IP01 · 05

네 개의 작업 view와 하나의 task detail이 서로 다른 질문에 답합니다

Board는 흐름, List는 비교, Activity는 변경, Evidence는 검증 상태를 보여 줍니다.

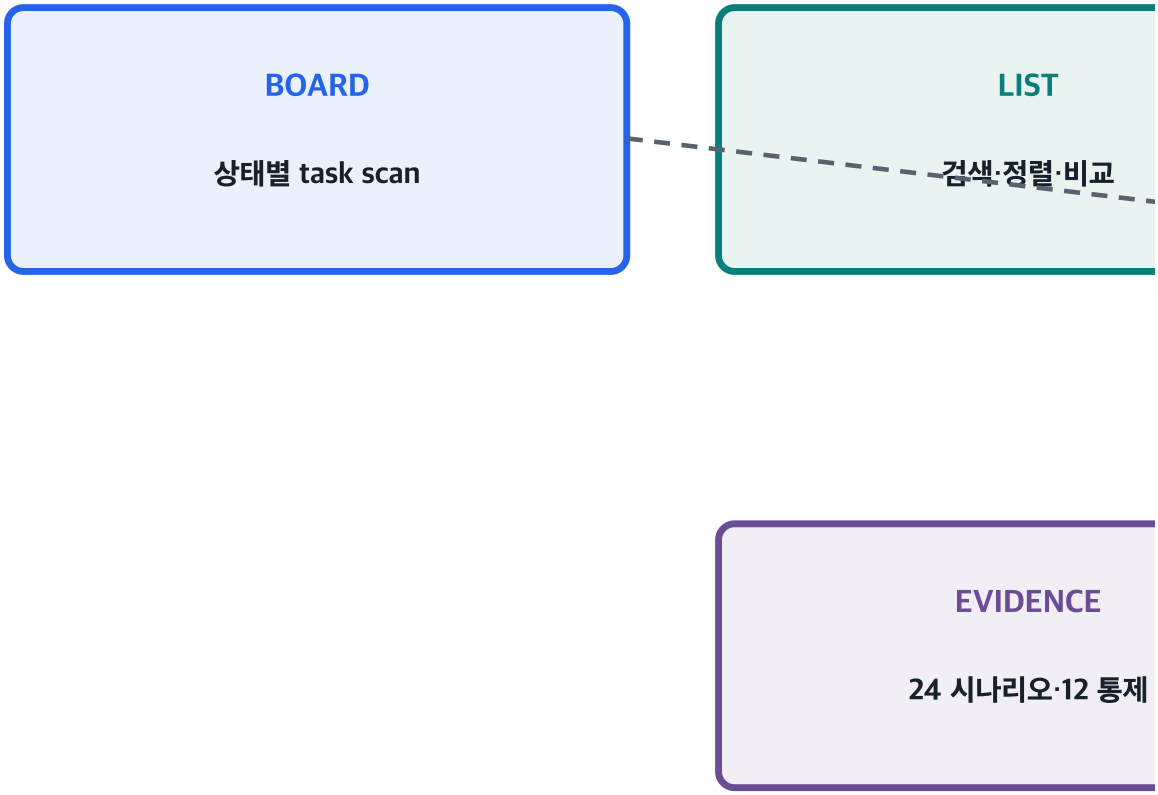


화면은 메뉴 수로 나누지 않고 사용자가 지금 내려야 할 판단과 행동을 기준으로 나눕니다.

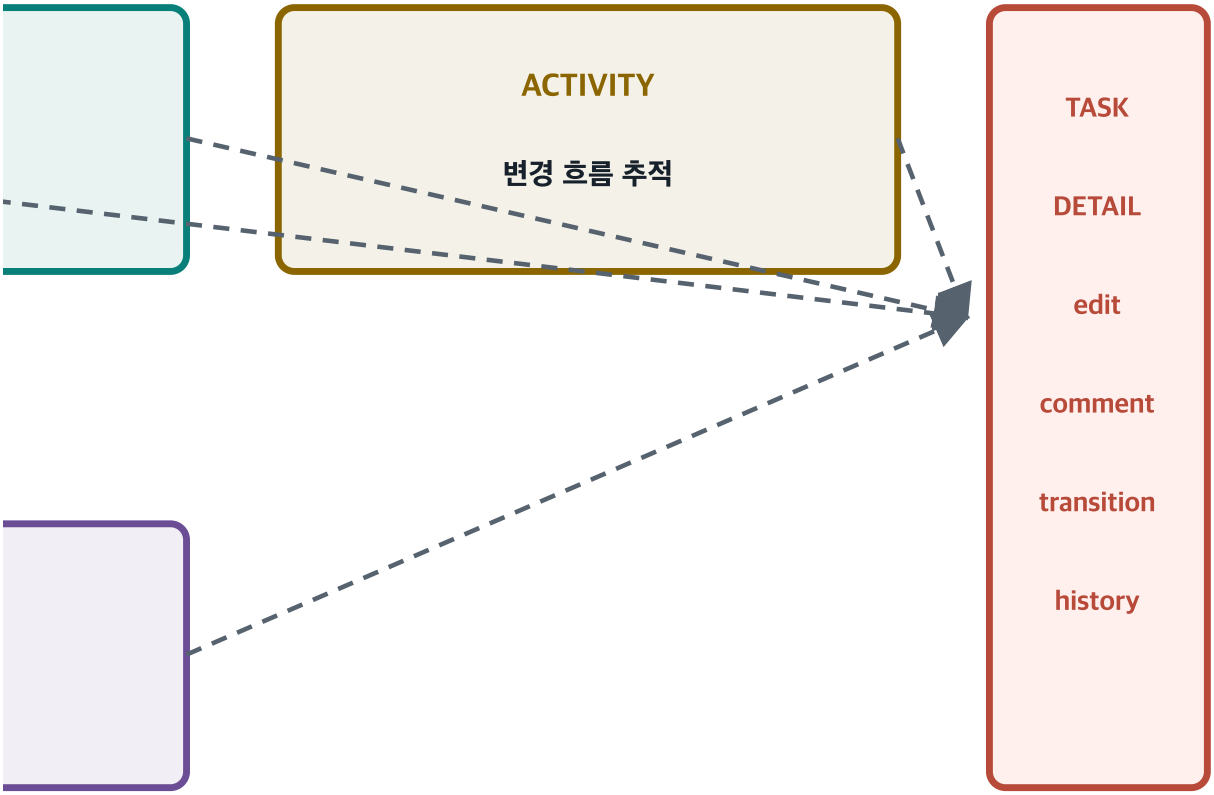
그림 5. 네 view는 각각 흐름·비교·변경·검증 질문에 답합니다.

세 개의 뷰가 서로 다른 task detail을

Board는 흐름, List는 비교, Activity는 변경, Evidence는 검증 상태를 보여 줍니



니다.


ACTIVITY
변경 흐름 추적
TASK
DETAIL
edit
comment
transition
history

핵심 원리
화면마다 사용자가 내려야 할 판단과 다음 행동이 하나 이상 분명해야 합니다.

Board는 어느 상태에 일이 몰렸는지 보고, List는 업무를 검색·비교하며, Activity는 누가 무엇을 바꿨는지 추적하고, Evidence는 프로젝트 검증 공백을 봅니다. Task detail은 어느 view에서도 같은 task의 계약·댓글·상태 전이·이력으로 들어가는 공통 작업 공간입니다.

관점	IP01에서 보이는 것	확인 evidence
Board	흐름과 병목은 어디인가	상태 열·카드
List	어떤 업무를 비교할 것인가	행·검색·정렬
Activity	무엇이 언제 바뀌었나	event timeline
Evidence	무엇이 확인되고 남았나	controls·scenarios
Task detail	이 업무의 다음 행동은 무엇인가	contract·comment·transition

흔한 오해
기능 이름을 그대로 메뉴로 만들고 같은 정보를 여러 화면에 반복합니다.

더 나은 판단
각 view가 답할 질문·주요 정보·가능한 명령·빈 상태를 한 줄로 고정합니다.

4단계 미니 실습

1. 네 view의 핵심 질문을 씁니다.
2. 중복 정보를 하나의 source로 정리합니다.
3. task detail로 들어가는 경로를 확인합니다.
4. empty·loading·error·success 상태를 붙입니다.

6. 업무 상태를 허용 transition으로 모델링하기

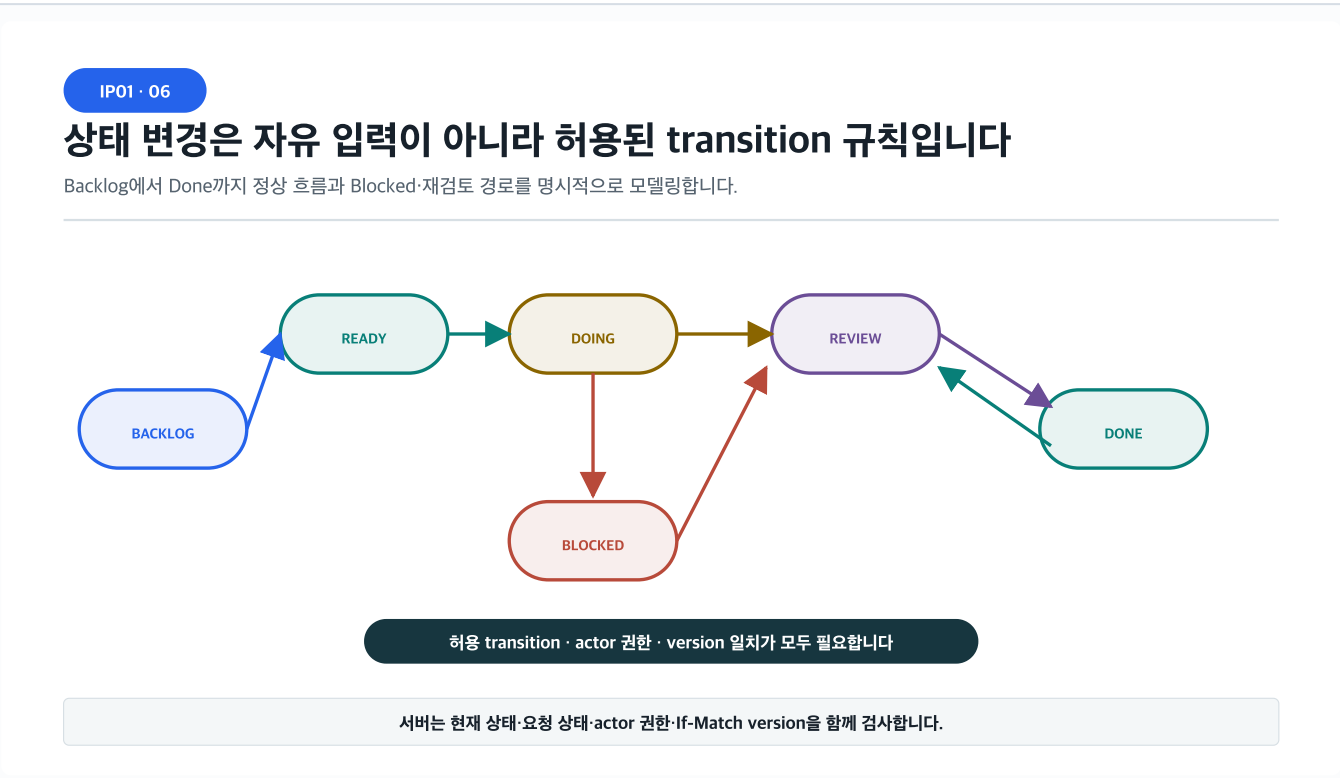
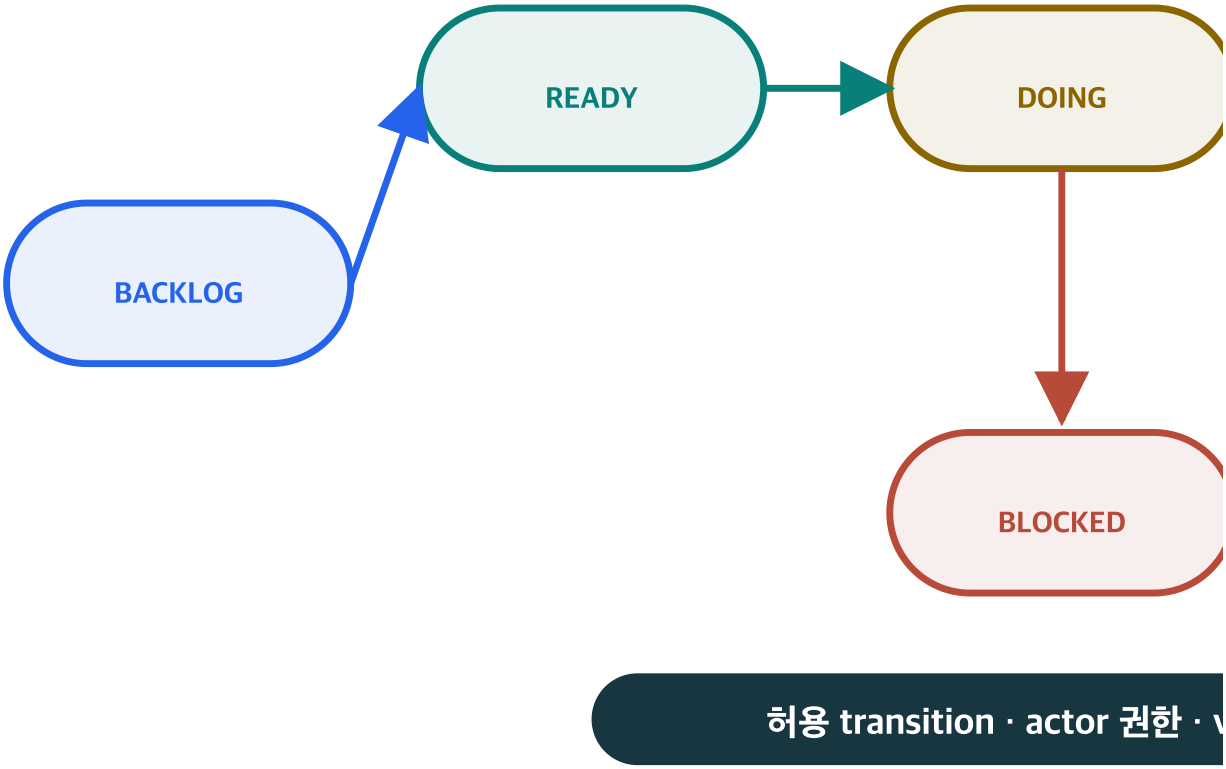


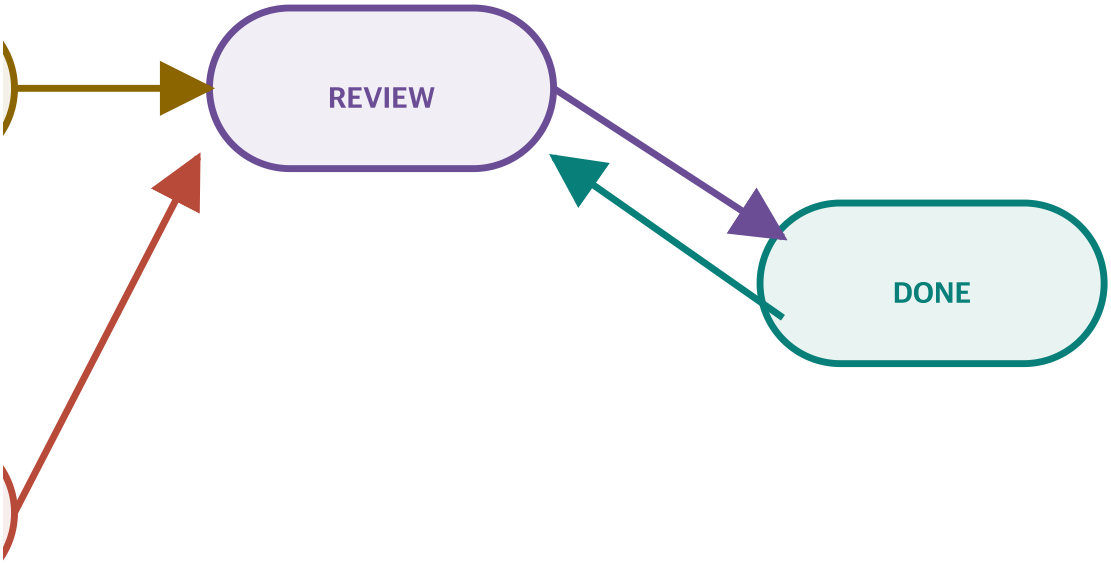
그림 6. 자유로운 상태 입력 대신 현재 상태와 규칙에 맞는 다음 상태만 허용합니다.

Backlog에서 Done까지 정상 흐름과 Blocked·재검토 경로를 명시적으로 모델



ransition 규칙 구성

일합니다.



ersion 일치가 모두 필요합니다

핵심 원리
상태 이름보다 중요한 것은 누가 어떤 조건에서 어디로 이동하고 무엇을 남기는지입니다.

Backlog→Ready→Doing→Review→Done이 기본 흐름이고, 의존 문제가 생기면 Blocked로 이동합니다. Done에는 완료 evidence가 필요하고 오래된 version에서 transition을 시도하면 충돌로 막힙니다. UI의 선택지도 서버가 계산한 허용 상태를 반영하지만 최종 판정은 domain rule이 담당합니다.

관점	IP01에서 보이는 것	확인 evidence
Backlog	요건 정리 중	Ready
Ready	시작 조건 충족	Doing·Blocked
Doing	실행 중	Blocked·Review
Blocked	진행 불가	Ready·Doing
Review/Done	검토·완료	evidence·reopen rule

흔한 오해
클라이언트가 원하는 상태 문자열을 보내면 데이터베이스에 그대로 저장합니다.

더 나은 판단
현재 상태·요청 상태·actor 권한·version·완료 evidence를 domain에서 함께 검사합니다.

4단계 미니 실습

1. 여섯 상태의 진입 조건을 적습니다.
2. 허용되지 않는 transition 세 개를 찾습니다.
3. Done에 필요한 evidence를 정합니다.
4. reopen과 Blocked 해제 흐름을 시험합니다.

7. 같은 사건을 UI·API·DB·evidence에서 추적하기

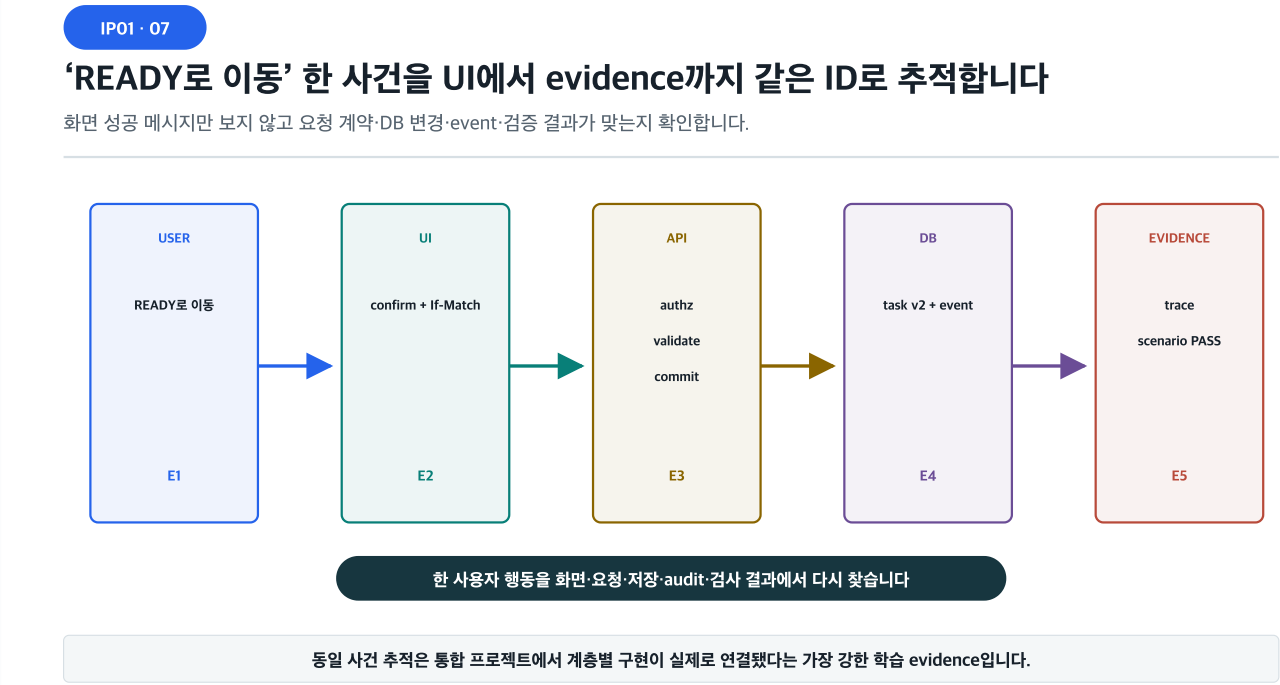
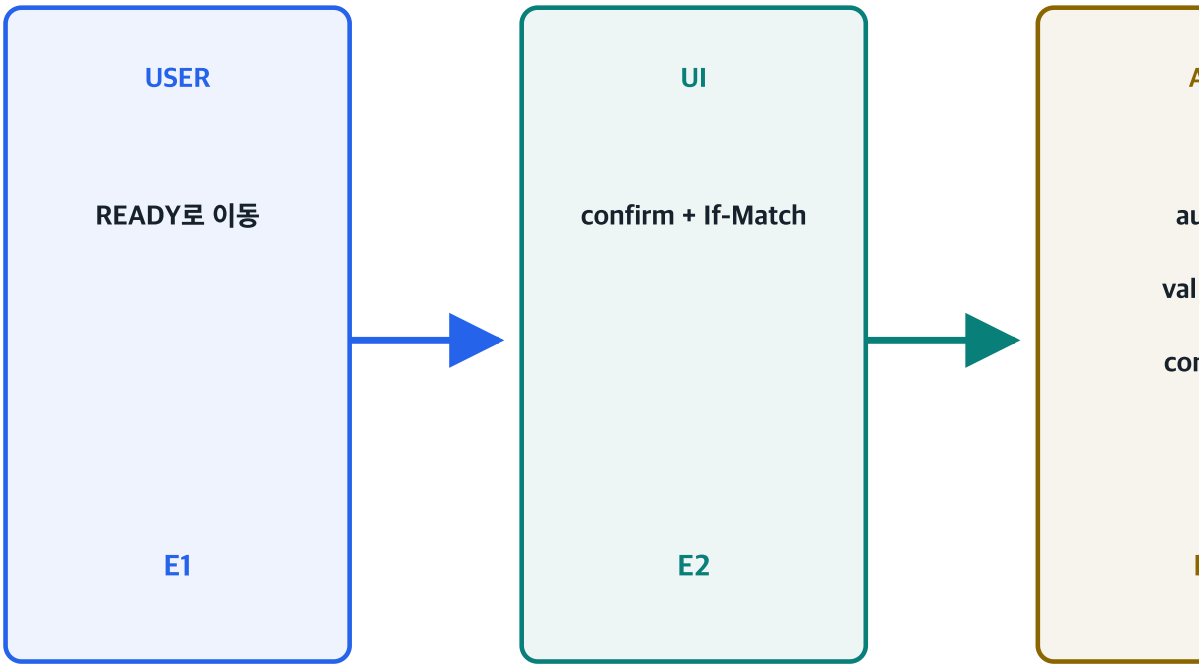


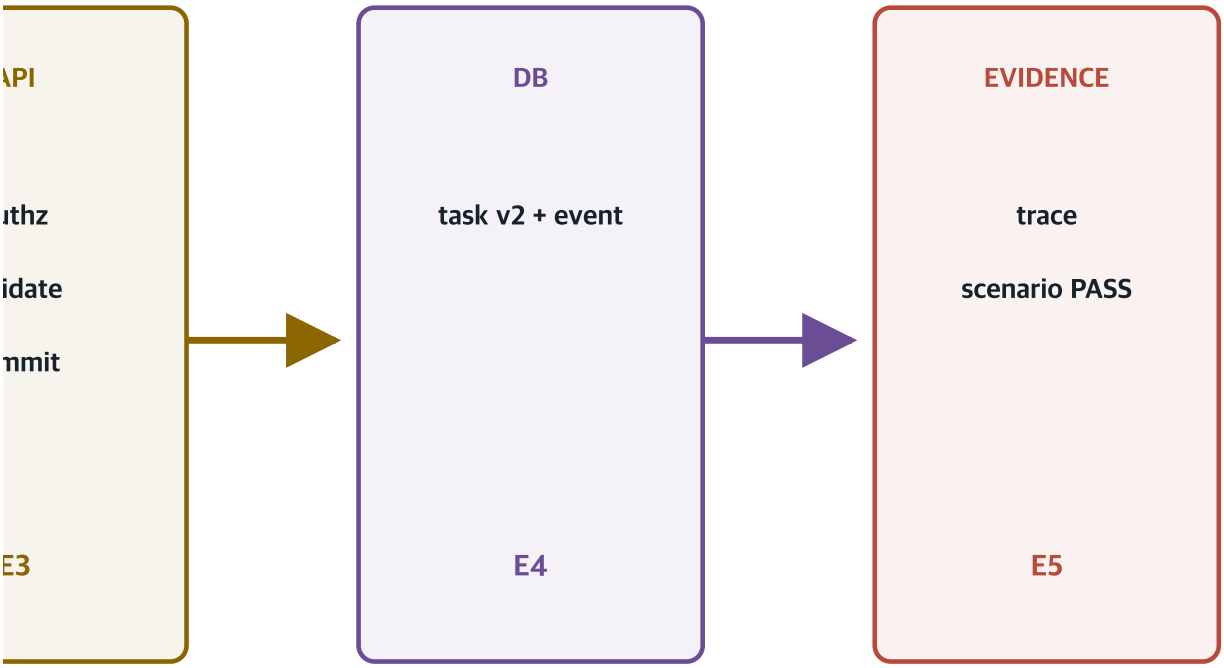
그림 7. ‘READY로 이동’ 한 행동을 계층마다 같은 task와 request 식별자로 찾습니다.

READY로 이동 후 다음을 어떻게 evidence

화면 성공 메시지만 보지 않고 요청 계약·DB 변경·event·검증 결과가 맞는지 확



인합니다.



udit·검사 결과에서 다시 찾습니다

핵심 원리
화면 성공 메시지와 실제 저장·audit·시험 결과가 같은 사건을 말해야 합니다.

사용자가 TASK-008을 Ready로 이동하면 UI는 현재 ETag를 If-Match로 보내고, API는 session·권한·입력·transition을 검사하며, transaction은 task version과 event를 함께 저장합니다. 응답의 새 version, Activity의 사건, audit의 request ID, 시나리오 결과가 서로 맞아야 통합됐다고 말할 수 있습니다.

관점	IP01에서 보이는 것	확인 evidence
User	상태 이동 명령	TASK-008
UI	If-Match·reason	request ID
API	authz·validate·transition	status·problem
DB	task v2·event	transaction
Evidence	audit·SC-12/16/17	expected=actual

흔한 오해
UI·API·DB 테스트가 각각 통과했다는 합계만 보고 실제 연결은 확인하지 않습니다.

더 나은 판단
한 사건을 골라 입력·식별자·version·시각·actor·결과를 계층별로 나란히 비교합니다.

4단계 미니 실습

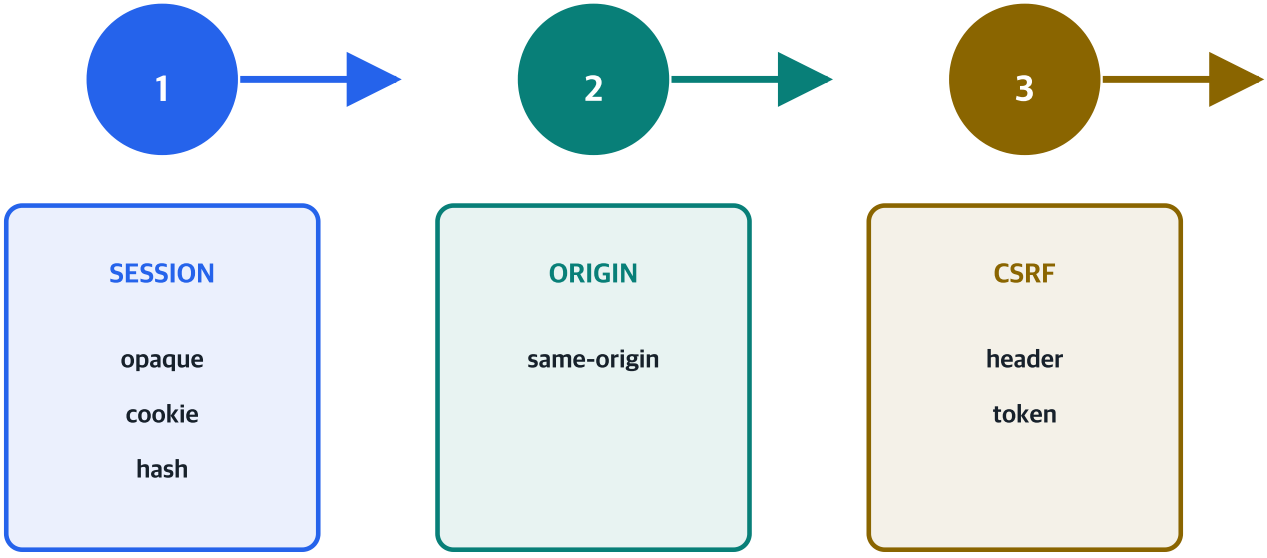
1. TASK-008 transition을 실행합니다.
2. 요청과 응답의 version을 적습니다.
3. DB event와 audit에서 같은 사건을 찾습니다.
4. 관련 시나리오의 expected와 actual을 대조합니다.

8. session부터 commit까지 보호 순서 설계하기

그림 8. 쓰기 요청은 여섯 보호 단계를 순서대로 통과합니다.

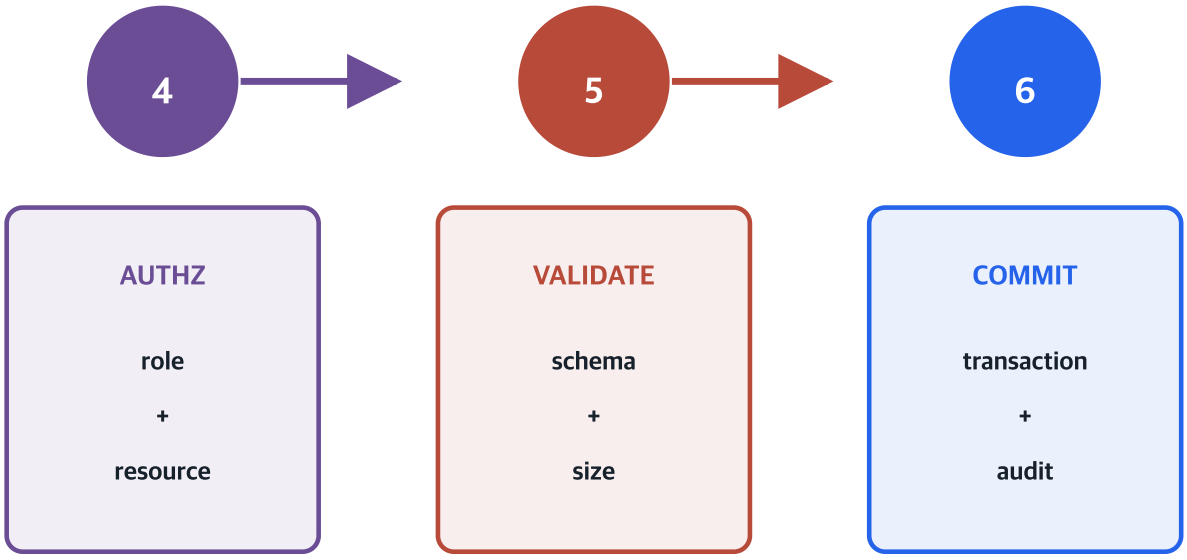
[illegible]

한 가지 보안 장치에 기대지 않고 요청의 출처·위조·권한·형식·저장을 층별로 검



앞 단계가 실패하면 이후 작업 없이

사합니다.



problem response로 종료합니다

핵심 원리
보안은 한 기능이 아니라 요청이 저장되기 전 여러 독립 판단이 겹치는 순서입니다.

IP01 session token은 브라우저 cookie에 있고 서버에는 hash만 저장합니다. 쓰기 요청은 session 유효성, same-origin, CSRF token, role·resource authorization, schema·body size validation을 통과한 뒤 transaction과 redacted audit를 남깁니다. 앞 단계가 실패하면 뒤의 mutation은 실행되지 않습니다.

관점	IP01에서 보이는 것	확인 evidence
Session	opaque token hash	만료·actor
Origin	same-origin	교차 출처 거부
CSRF	header token	위조 요청 거부
Authorization	role·resource	allow/deny
Validate/commit	schema·size·transaction	audit result

흔한 오해
HttpOnly cookie가 있으므로 모든 쓰기 요청이 안전하다고 봅니다.

더 나은 판단
session·origin·CSRF·authorization·validation·transaction을 각각 독립 실패 시나리오로 검사합니다.

4단계 미니 실습

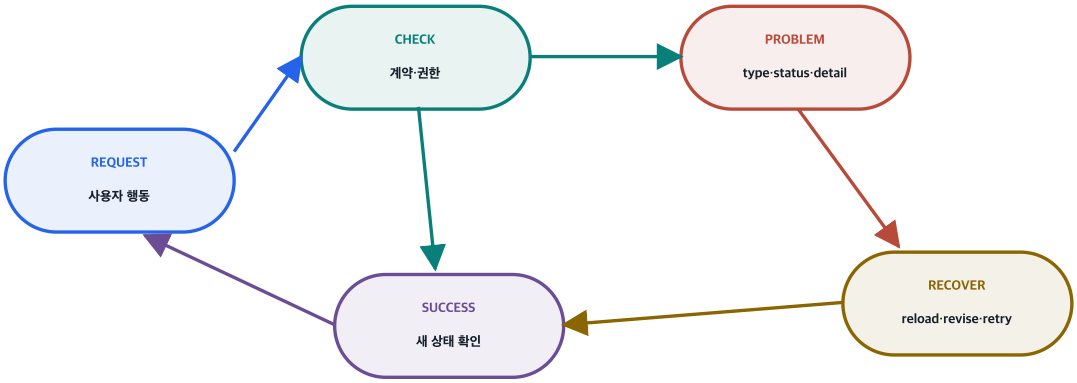
1. 쓰기 요청의 여섯 검사를 순서대로 적습니다.
2. 각 단계에서 실패할 예를 하나씩 만듭니다.
3. 실패 뒤 DB가 바뀌지 않았는지 확인합니다.
4. audit에 token 원문이 없는지 검사합니다.

9. 오류를 복구 가능한 Problem 계약으로 만들기

IP01 · 09

오류는 막다른 문장이 아니라 다음 행동을 알려 주는 복구 계약입니다

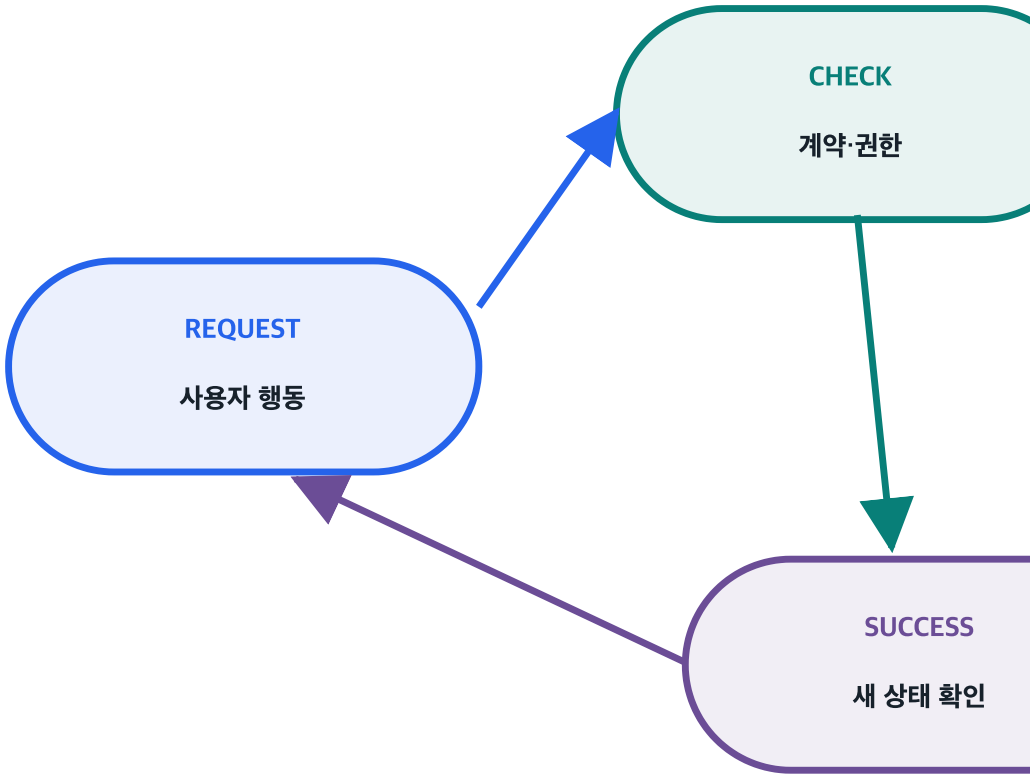
Problem Details 응답을 화면 메시지·필드 수정·reload·retry 동작과 연결합니다.



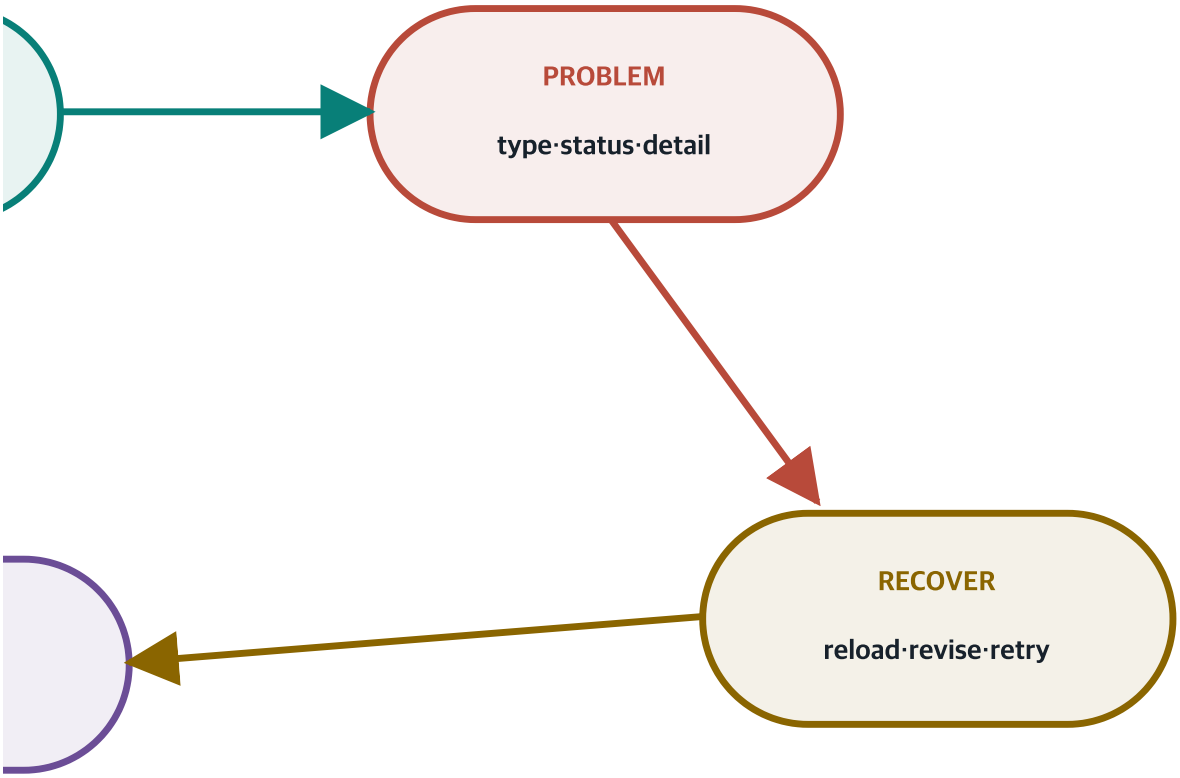
좋은 오류 처리는 실패를 숨기지 않고 원인·영향·사용자가 할 수 있는 다음 행동을 구분해 보여 줍니다.

그림 9. 오류 type·detail을 필드 수정·reload·retry 행동과 연결합니다.

Problem Details 응답을 화면 메시지·필드 수정·reload·retry 동작과 연결합니



다.



핵심 원리
좋은 오류는 실패를 감추지 않고 원인·영향·사용자가 할 수 있는 다음 행동을 구분합니다.

서버는 RFC 9457 형식의 `application/problem+json`으로 type·title·status·detail·instance와 field 오류를 반환합니다. UI는 이를 '실패했습니다' 한 문장으로 뭉개지 않고 입력 수정, 최신 상태 reload, 권한 확인, 나중 재시도 중 알맞은 복구 행동으로 바꿉니다.

관점	IP01에서 보이는 것	확인 evidence
400	요청 형식 문제	입력 구조 확인
403	권한·출처 거부	역할·session 확인
409	상태·version 충돌	reload 후 의도 재확인
422	field validation	해당 입력 수정
5xx/connection	서버·연결 문제	상태 확인·안전한 retry

흔한 오해
모든 오류를 alert('오류')로 보여 주고 사용자가 다시 처음부터 시도하게 합니다.

더 나은 판단
problem type별 화면 메시지·보존할 입력·자동/수동 복구·재시도 가능성을 명시합니다.

4단계 미니 실습

1. validation 오류를 일부러 만듭니다.
2. 응답 type·status·detail을 기록합니다.
3. 화면이 알려 주는 다음 행동을 확인합니다.
4. 수정 뒤 새 상태가 반영됐는지 검증합니다.

10. 중복 요청과 동시 수정 충돌을 따로 해결하기

그림 10. 반복 생성은 Idempotency-Key로, 오래된 수정은 ETag·If-Match로 다룹니다.

Idempotency-Key는 반복 요청을, ETag·If-Match는 오래된 version 덮어쓰기

중복 방지와 충돌 감지는 서

를 막습니다.

수정

RETRY

최신 상태로 재시도

reload v2

의도 다시 확인

commit v3

로 다른 실패를 해결합니다

핵심 원리
중복 방지와 충돌 감지는 원인도 복구 행동도 다른 계약입니다.

네트워크 재시도로 새 업무 POST가 두 번 도착할 수 있습니다. 같은 Idempotency-Key면 첫 결과를 재사용해 task를 하나만 만듭니다. 반면 두 사용자가 같은 task를 수정하면 오래된 If-Match를 보낸 요청을 409로 거부하고 최신 version을 다시 읽게 합니다.

관점	IP01에서 보이는 것	확인 evidence
Duplicate	같은 POST 반복	Idempotency-Key
Store	key+첫 결과	중복 task 0
Stale edit	ETag v1로 수정	If-Match v1
Conflict	현재 v2	409 problem
Recover	reload v2·의도 확인	commit v3

흔한 오해
오류가 나면 아무 요청이나 자동으로 여러 번 다시 보냅니다.

더 나은 판단
명령 종류별 중복 위험을 판단하고 key 재사용 범위와 version 충돌 복구 UI를 설계합니다.

4단계 미니 실습

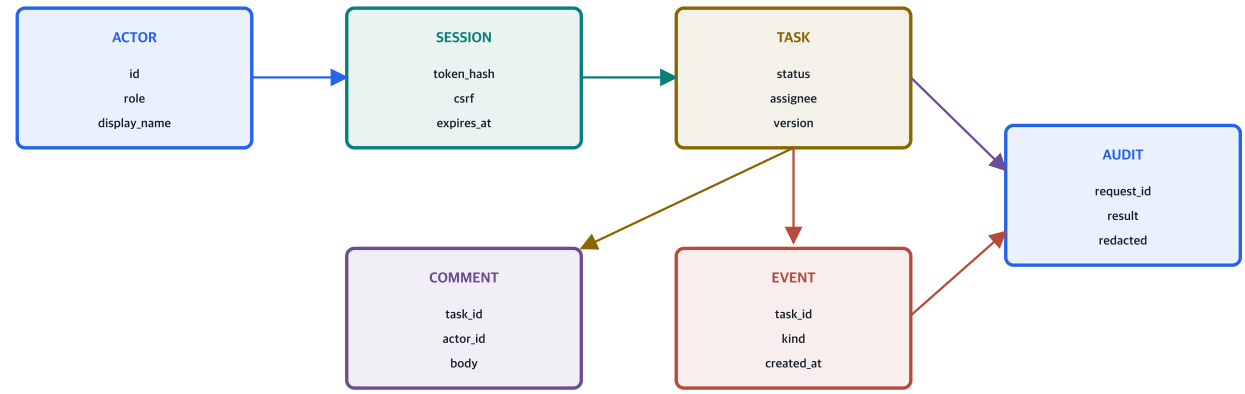
1. 같은 key로 create를 두 번 보냅니다.
2. 생성된 task 수를 확인합니다.
3. 오래된 version으로 edit를 시도합니다.
4. 409 뒤 reload·재시도 흐름을 기록합니다.

11. 현재 상태와 변경 이력을 다른 데이터로 저장하기

IP01 · 11

task를 중심으로 actor·session·comment·event·audit 관계를 나눕니다

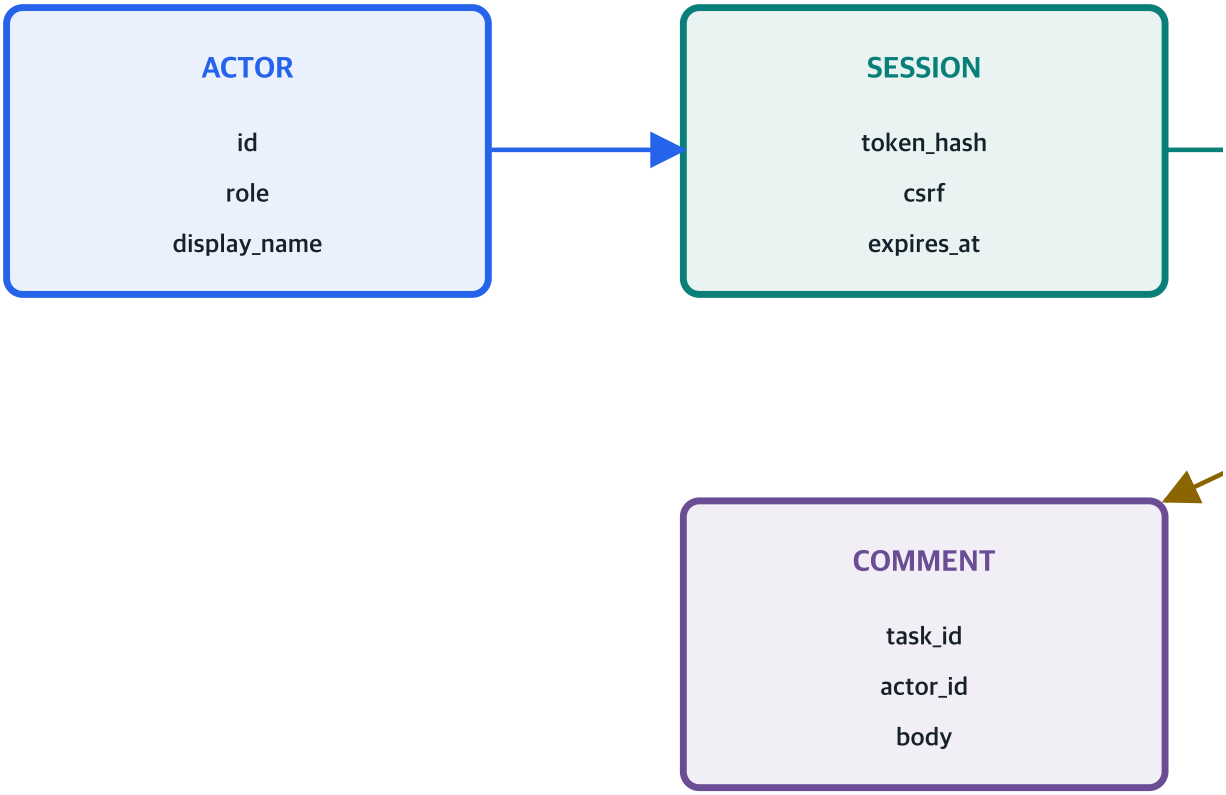
현재 상태와 변경 이력, 보안 session, 운영 trace를 서로 다른 책임의 데이터로 저장합니다.



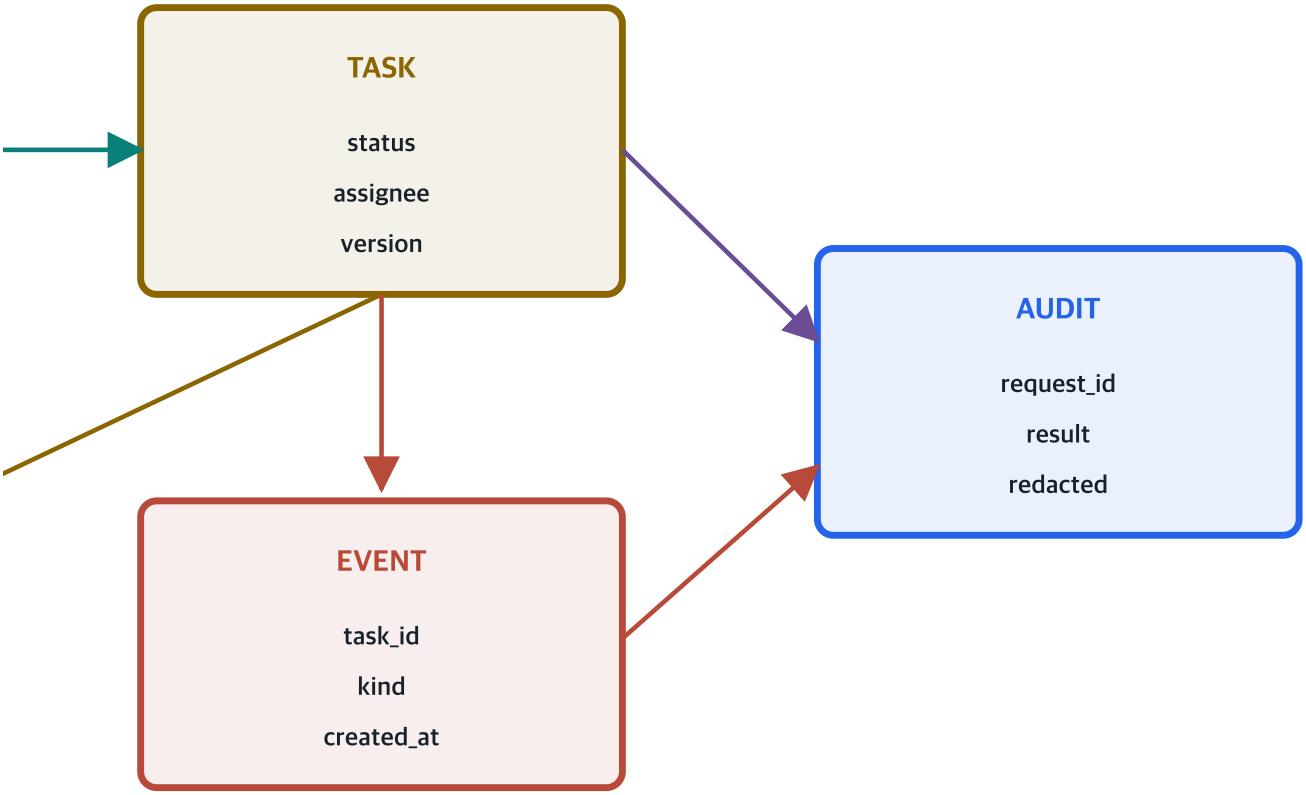
SQLite는 이 로컬 학습 범위에 맞는 선택이며 다중 서버·대규모 동시성·외부 identity 요구를 해결하지 않습니다.

그림 11. task 현재 상태와 comment·event·audit 이력을 책임별 테이블로 나눕니다.

현재 상태와 변경 이력, 보안 session, 운영 trace를 서로 다른 책임의 데이터로



저장합니다.


TASK
status
assignee
version
AUDIT
request_id
result
redacted
EVENT
task_id
kind
created_at

핵심 원리
한 task 행에 모든 것을 덧붙이지 말고 identity·관계·변경 이력·운영 기록의 책임을 분리합니다.

Actor는 역할을, Session은 token hash와 CSRF를, Task는 현재 상태와 version을, Comment는 의견을, Event는 업무 변경을, Audit는 요청 판정을 보존합니다. Foreign key·unique·check·not-null 제약과 transaction이 애플리케이션 검사를 보완합니다.

관점	IP01에서 보이는 것	확인 evidence
Actor	id·role·name	permission subject
Session	token_hash·csrf·expiry	actor relation
Task	status·assignee·version	current state
Comment/Event	task relation·content/kind	collaboration·history
Audit	request·actor·result	redacted operational trace

흔한 오해
JSON 한 덩어리에 task·댓글·이력·session을 모두 저장하고 관계 검사를 코드에만 둡니다.

더 나은 판단
현재 상태·업무 이력·보안 session·운영 audit를 분리하고 DB 제약과 transaction을 시험합니다.

4단계 미니 실습

1. 여섯 entity의 primary key를 표시합니다.
2. foreign key 관계를 화살표로 그립니다.
3. 상태와 version 제약을 적습니다.
4. 부분 실패 때 rollback되는지 확인합니다.

12. 테스트를 수가 아니라 위험 portfolio로 구성하기

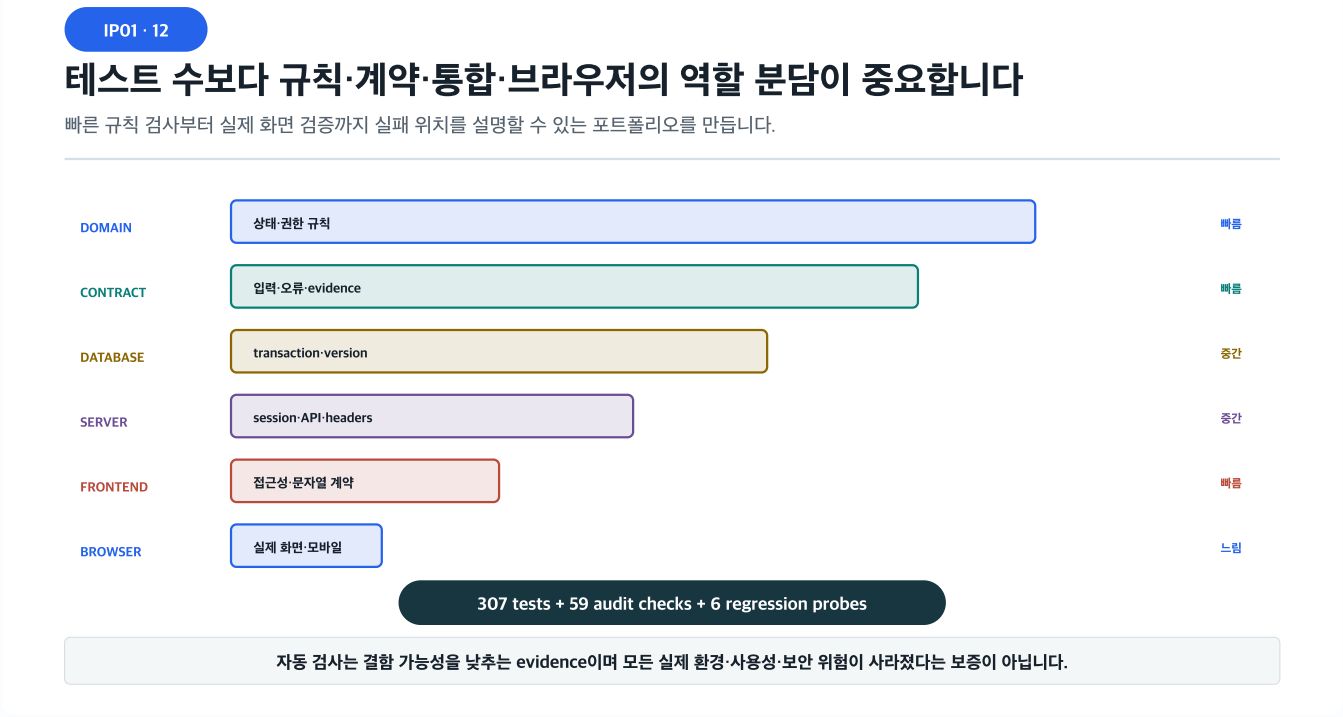
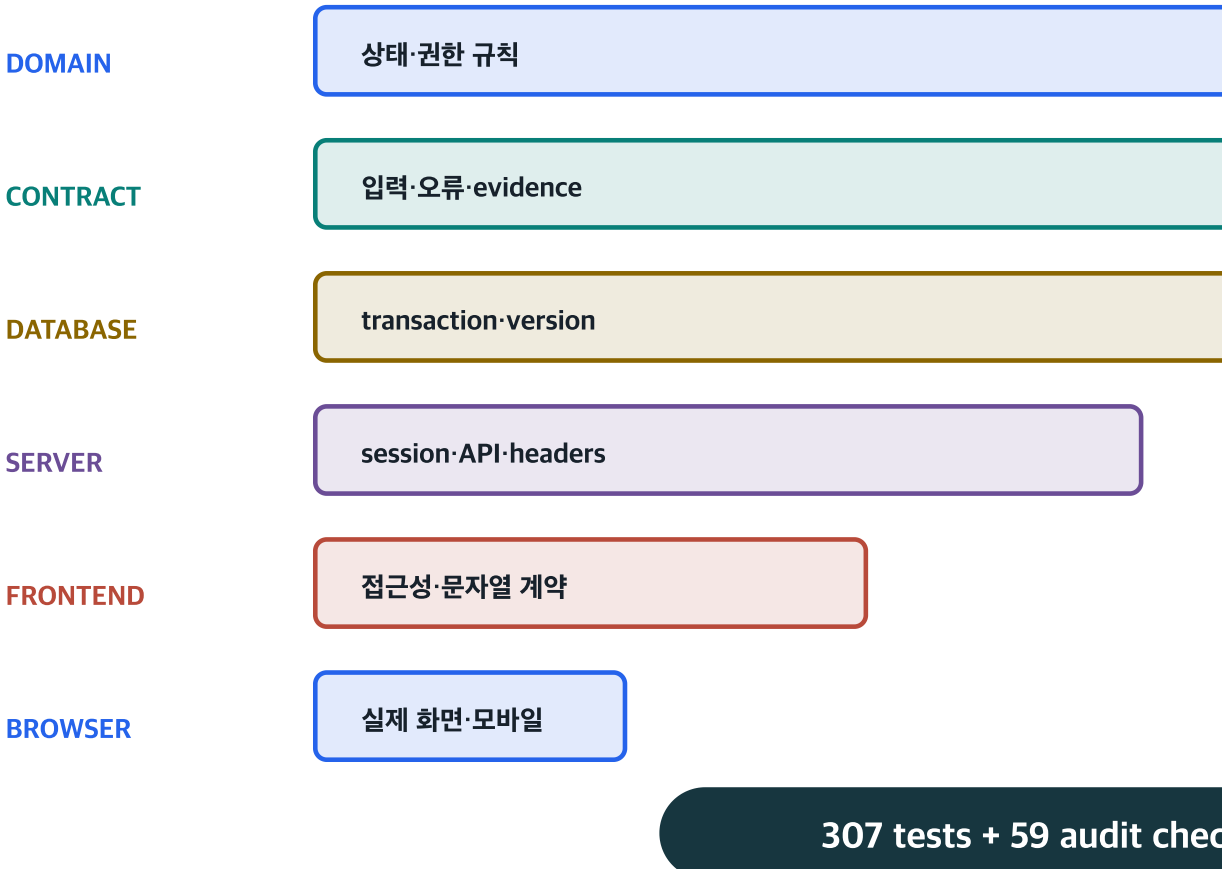


그림 12. 빠른 규칙 검사와 실제 브라우저 검증의 역할을 나눕니다.

빠른 규칙 검사부터 실제 화면 검증까지 실패 위치를 설명할 수 있는 포트폴리오



[illegible]

[illegible]를 만듭니다.

핵심 원리
307개라는 숫자보다 어느 위험을 어떤 종류의 시험이 설명하는지가 중요합니다.

Domain test는 상태·권한 규칙을 빠르게 잡고, contract test는 12개 통제·24개 시나리오 형식을 확인하며, database test는 제약·transaction·version을, server test는 session·API·headers를, frontend contract test는 접근성 구조와 위험한 DOM 사용을 검사합니다. 마지막에 실제 브라우저로 desktop·mobile·role 흐름을 확인합니다.

관점	IP01에서 보이는 것	확인 evidence
Domain	state·permission	빠른 규칙 결함
Contract	controls·scenarios·variants	evidence 공백
Database	constraint·transaction·version	무결성 결함
Server	session·API·security headers	통합·보안 결함
Frontend/Browser	DOM·a11y·responsive·workflow	사용자 경험 결함

흔한 오해
모든 경우를 느린 브라우저 시험으로 만들거나 unit test 수만 늘립니다.

더 나은 판단
실패 위치와 피드백 속도를 기준으로 규칙·계약·통합·브라우저 시험을 배치합니다.

4단계 미니 실습

1. 현재 시험을 다섯 층으로 분류합니다.
2. 각 critical scenario의 가장 가까운 시험을 찾습니다.
3. negative·boundary·recovery 검사를 표시합니다.
4. 자동 시험이 답하지 못하는 질문을 적습니다.

13. 관찰·복구·학습을 운영 loop로 연결하기

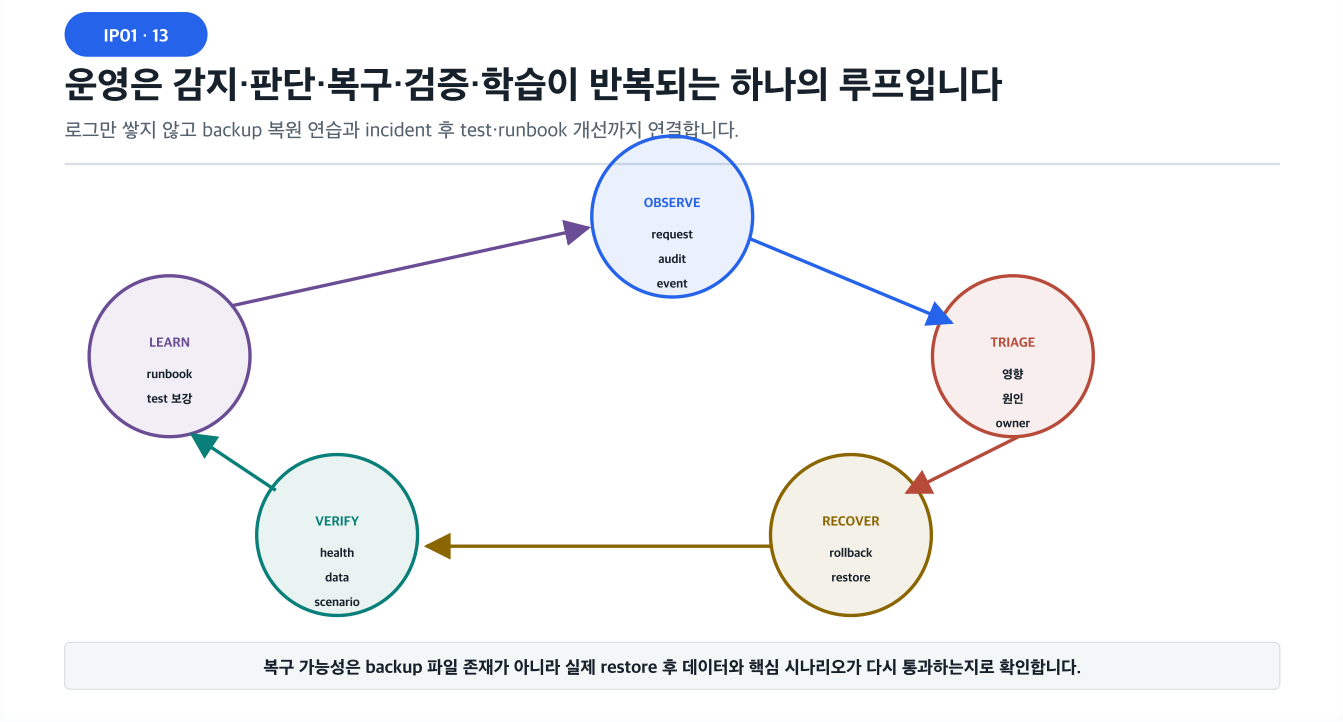
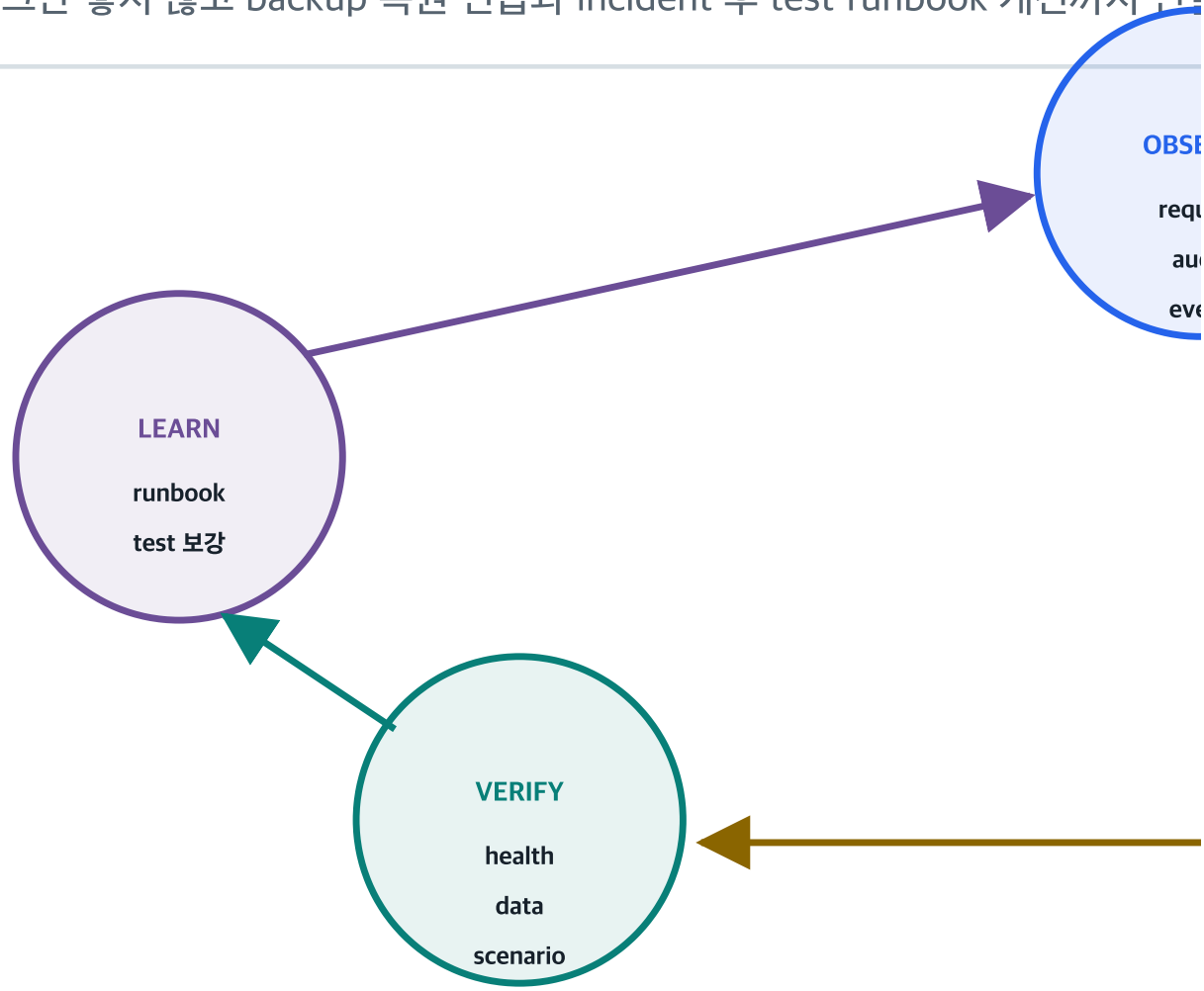
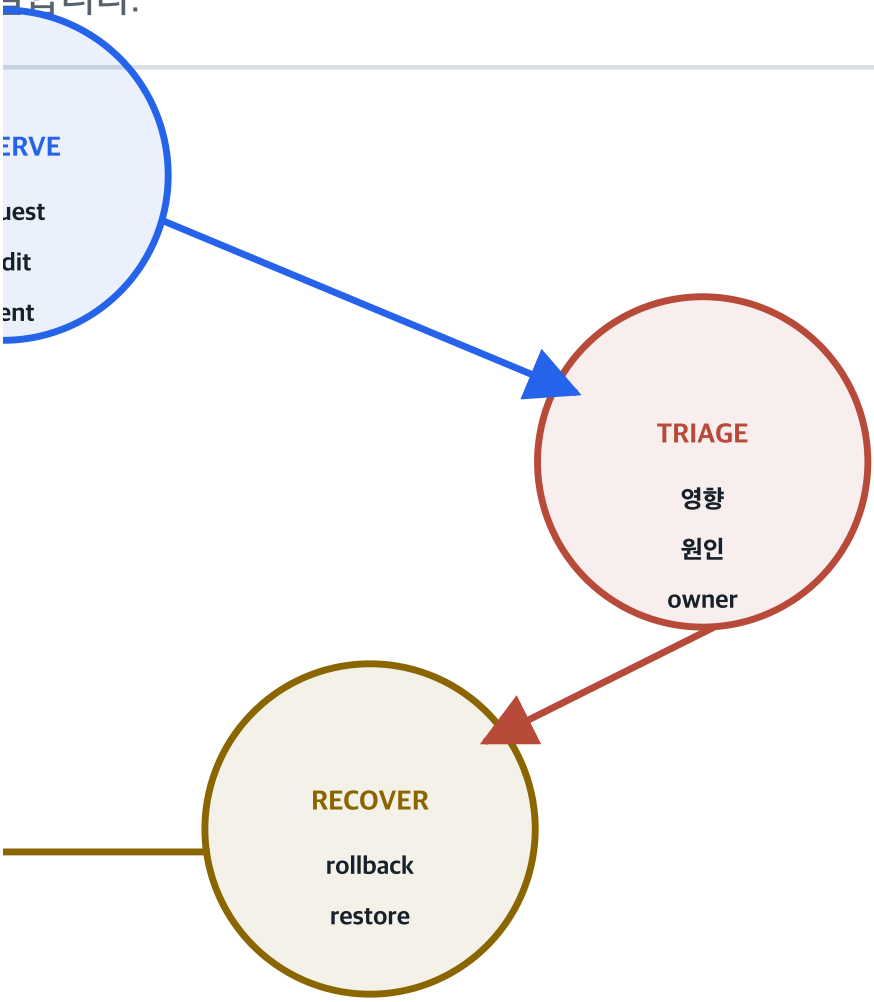


그림 13. request·audit 관찰에서 restore·검증·runbook 개선까지 한 바퀴를 돕니다.

로그만 쌓지 않고 backup 복원 연습과 incident 후 test·runbook 개선까지 연결



결합니다.


ERVE
uest
dit
ent
TRIAGE
영향
원인
owner
RECOVER
rollback
restore

핵심 원리
backup 파일이 있다는 것보다 실제로 복원하고 핵심 시나리오가 다시 통과하는지가 중요합니다.

운영자는 request ID와 audit·event로 영향을 관찰하고, incident를 triage한 뒤 rollback 또는 backup restore를 수행합니다. 복원 후 health·행 수·관계·핵심 사용자 흐름을 확인하고 발견한 공백을 test와 runbook에 반영해야 다음 사건의 복구 능력이 높아집니다.

관점	IP01에서 보이는 것	확인 evidence
Observe	health·request·audit·event	영향 탐지
Triage	범위·긴급도·원인 후보	owner·decision
Recover	rollback·restore	데이터·서비스 복구
Verify	integrity·health·scenario	새 상태 확인
Learn	test·runbook·design	재발 가능성 감소

흔한 오해
DB 파일을 복사해 두고 backup 검증 완료라고 표시합니다.

더 나은 판단
독립 경로로 restore하고 데이터 무결성·핵심 API·사용자 시나리오를 다시 실행한 evidence를 남깁니다.

4단계 미니 실습

1. snapshot을 만듭니다.
2. 합성 변경 뒤 backup으로 복원합니다.
3. 행 수·task 상태·health를 확인합니다.
4. 발견한 개선점을 test나 runbook에 추가합니다.

14. 실행물·evidence·학습 문서·인계를 한 package로 만들기

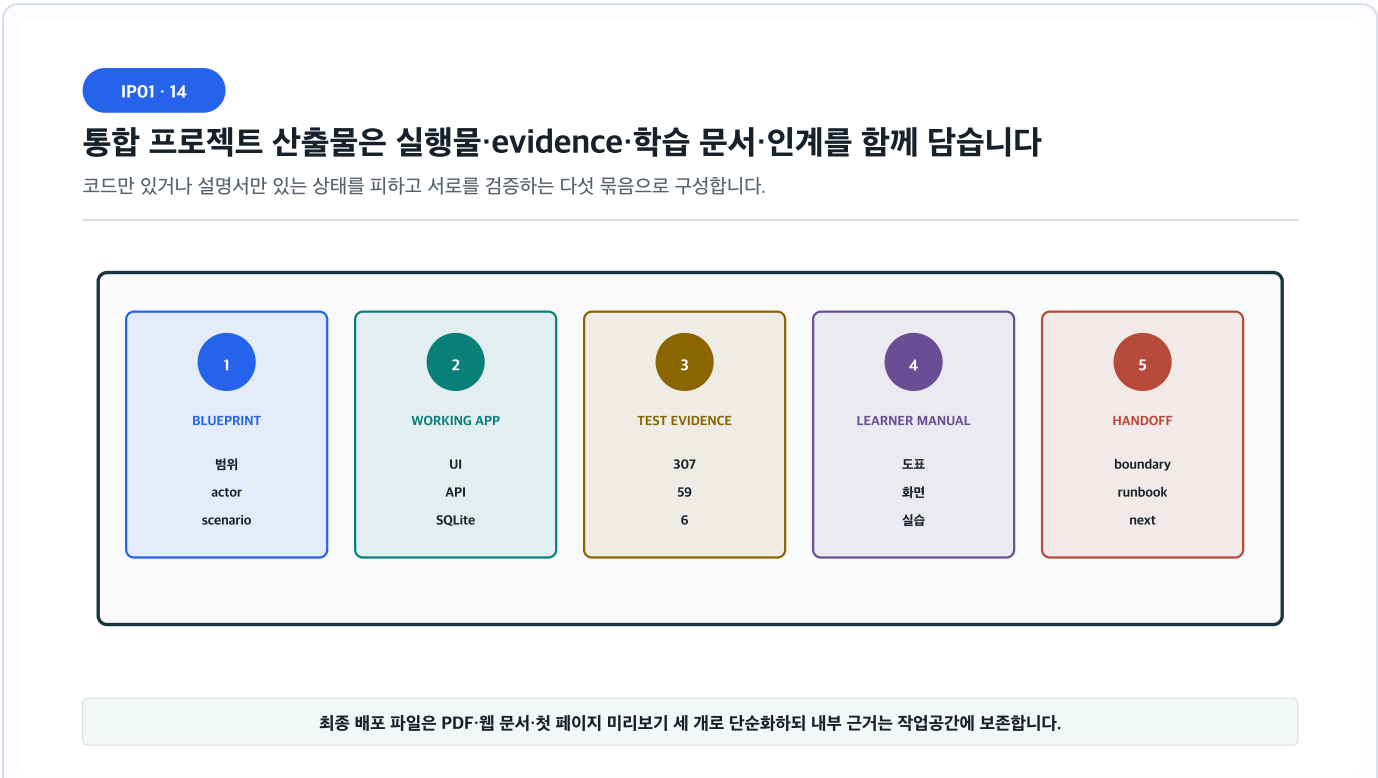
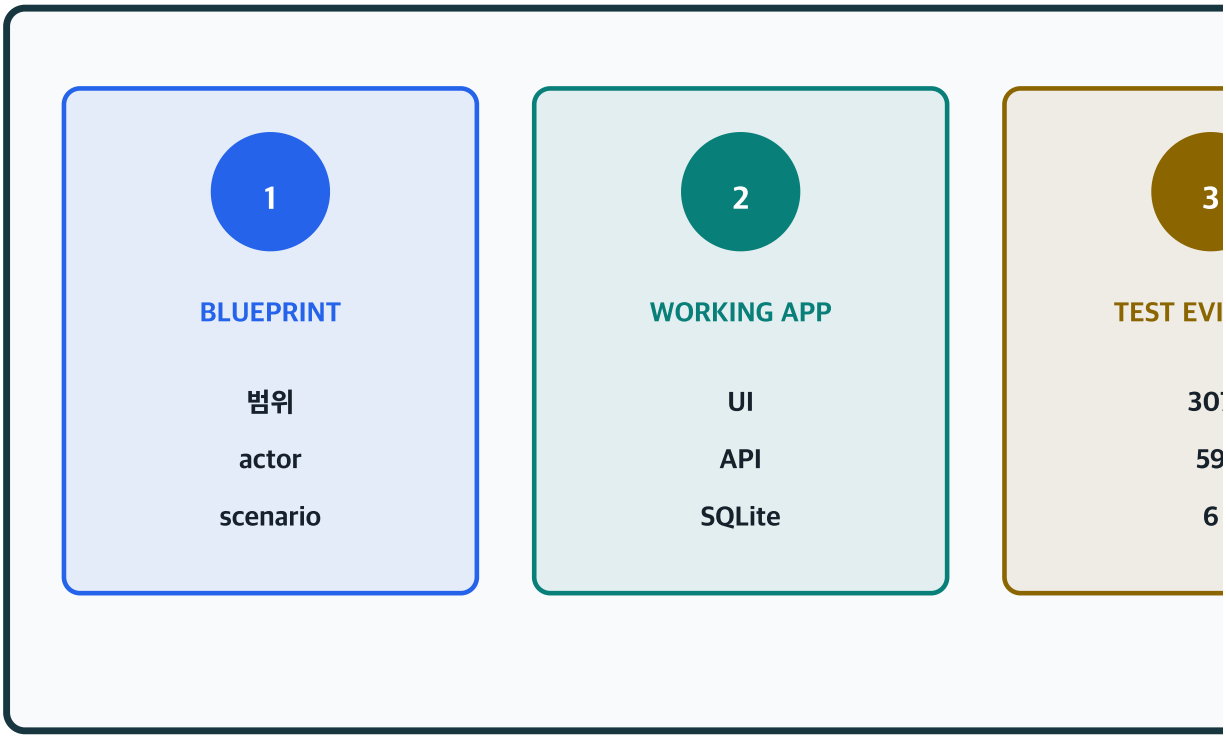


그림 14. 프로젝트 결과를 다섯 묶음으로 나눠 서로 검증하게 합니다.

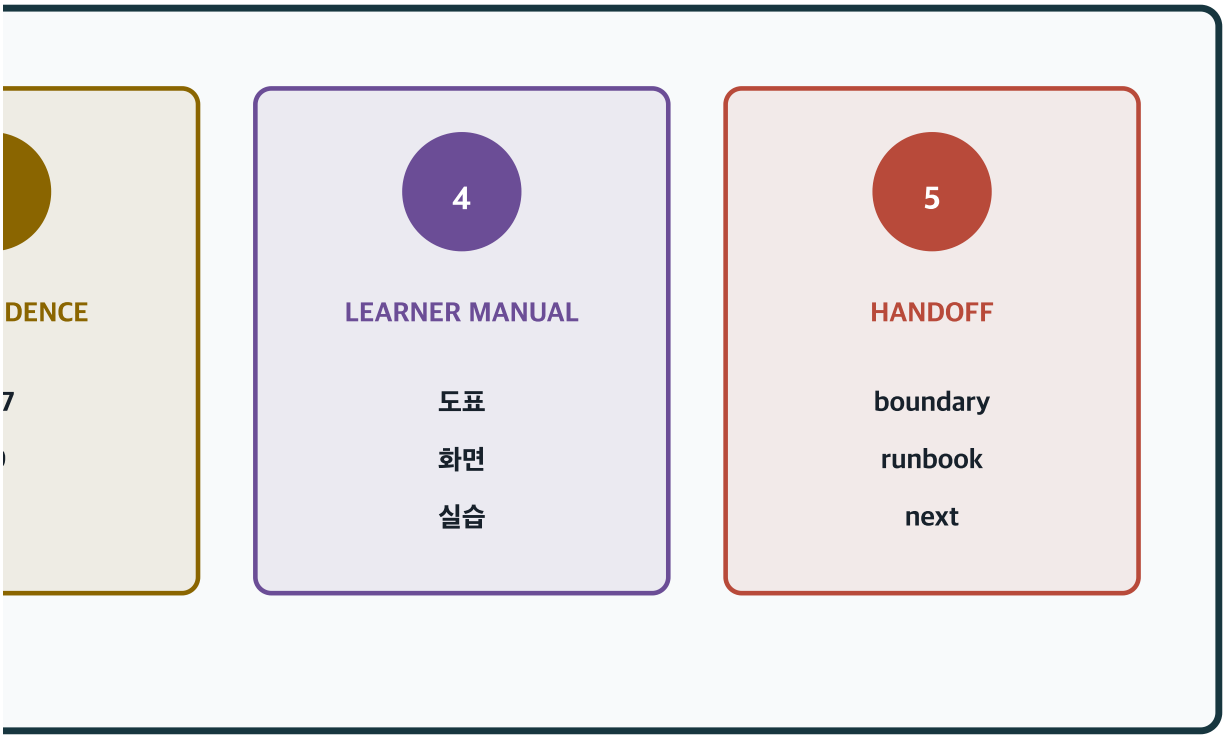
[illegible]

코드만 있거나 설명서만 있는 상태를 피하고 서로를 검증하는 다섯 묶음으로 구



[illegible]

·성합니다.


DENCE
7
4
LEARNER MANUAL
도표
화면
실습
5
HANDOFF
boundary
runbook
next

핵심 원리
코드만 있거나 설명서만 있는 상태를 피하고 실행 가능한 source와 재현 가능한 evidence를 함께 넘깁니다.

Blueprint는 문제·actor·시나리오를 고정하고, working app은 실제 흐름을 보여 주며, test evidence는 기대와 실제를 비교합니다. Learner manual은 복잡한 연결을 도표와 화면으로 설명하고, handoff는 실행·변경·복구·잔여 위험을 후속 담당자에게 넘깁니다. 최종 공개 파일은 PDF·웹·미리보기 세 개로 단순화합니다.

관점	IP01에서 보이는 것	확인 evidence
Blueprint	scope·actor·scenario	무엇을 왜 만드는가
Working app	UI·API·SQLite	실제로 동작하는가
Test evidence	307·59·6	규칙과 연결이 맞는가
Learner manual	15 diagrams·9 screens	학습자가 설명 가능한가
Handoff	runbook·risk·next	다른 사람이 이어갈 수 있는가

흔한 오해
최종 PDF에 구현 설명만 넣고 source·시험·복구 방법은 별도 담당자의 기억에 둡니다.

더 나은 판단
산출물마다 owner·version·생성 방법·검증 결과·authority boundary·next action을 연결합니다.

4단계 미니 실습

1. 다섯 묶음의 실제 경로를 적습니다.
2. 각 묶음의 owner와 version을 확인합니다.
3. 고아 artifact가 없는지 찾습니다.
4. 후속 담당자가 처음 실행할 절차를 적습니다.

15. Integrated Project Gate로 검토 준비 상태 판단하기

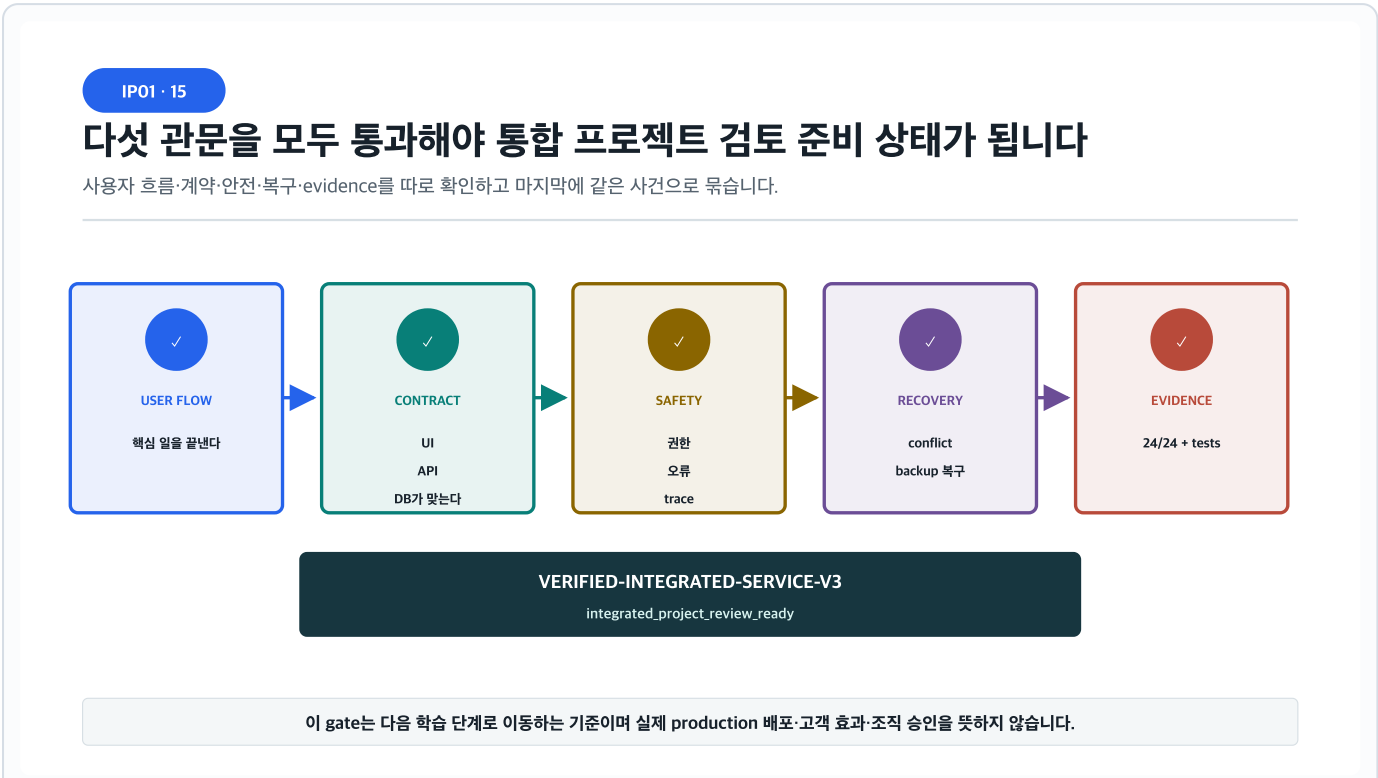
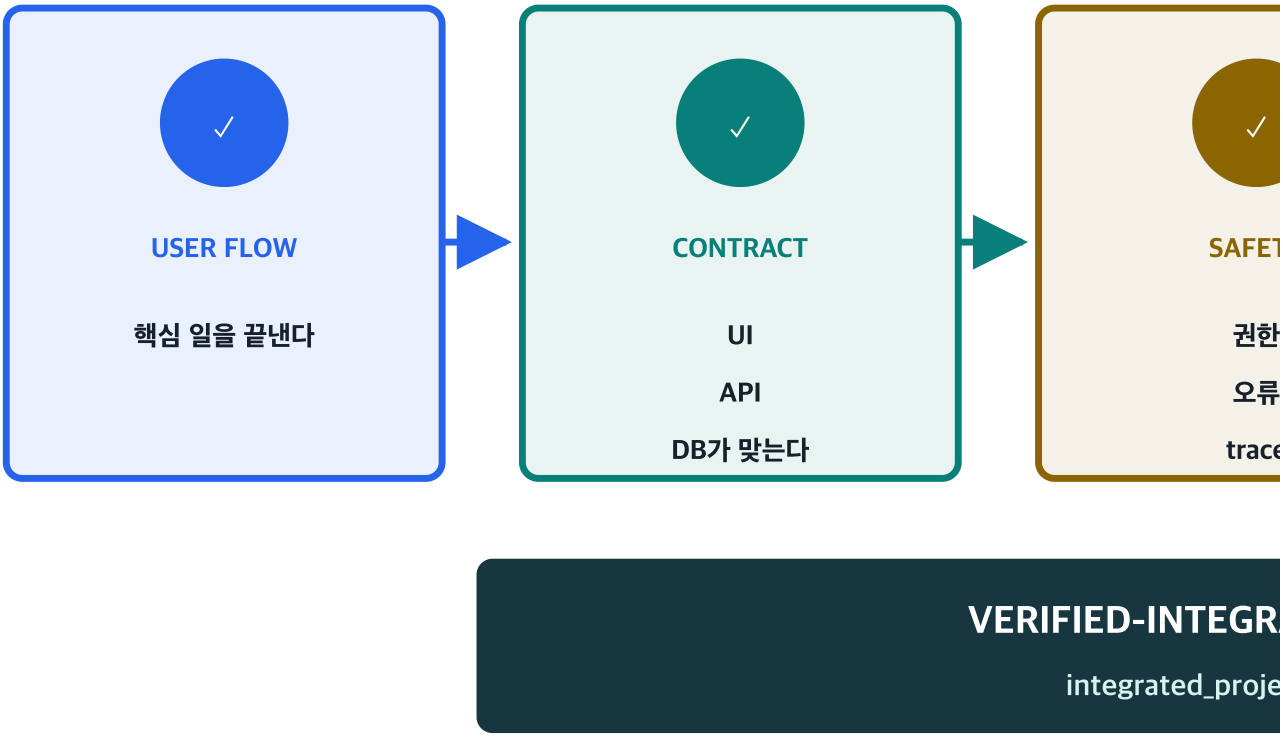


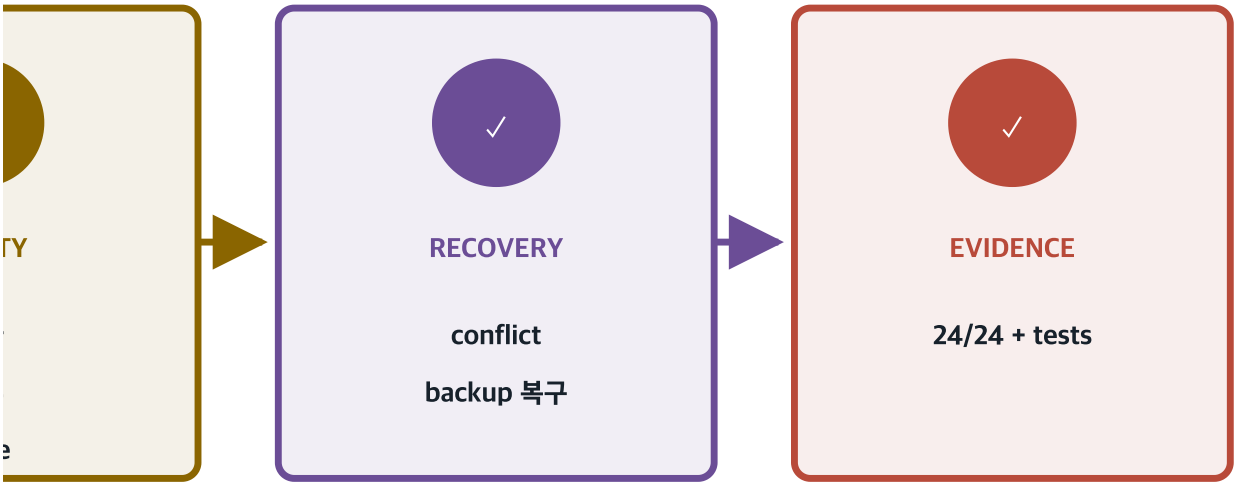
그림 15. 사용자 흐름·계약·안전·복구·evidence 다섯 관문을 모두 확인합니다.

사용자 흐름·계약·안전·복구·evidence를 따로 확인하고 마지막에 같은 사건으로



[illegible]

[illegible]를 묶습니다.



[illegible]ATED-SERVICE-V3

[illegible]ct_review_ready

핵심 원리
Gate는 다음 학습 단계로 이동할 기준이며 production 배포·고객 효과·조직 승인을 뜻하지 않습니다.

최종 후보는 24/24 scenario, critical 12/12, control 12/12, orphan 0, regression 0을 만족합니다. 하지만 숫자를 모으는 것이 목적은 아닙니다. 사용자가 핵심 일을 끝내고, UI·API·DB 계약이 맞으며, 권한·오류·trace가 있고, 충돌·backup에서 복구하며, 검토자가 이를 재현할 수 있어야 합니다.

관점	IP01에서 보이는 것	확인 evidence
User flow	핵심 일을 끝냄	happy+failure path
Contract	UI·API·DB 일치	same-event trace
Safety	permission·validation·audit	negative tests
Recovery	conflict·restore	rehearsal evidence
Evidence	24/24·307·59·6	review ready

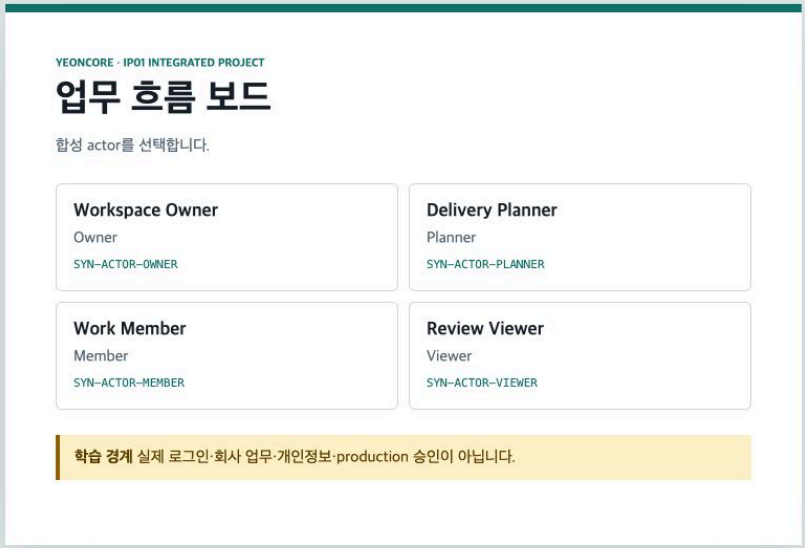
흔한 오해
한 번의 시연이 성공하면 모든 Gate를 통과한 것으로 표시합니다.

더 나은 판단
다섯 관문을 각각 확인하고 마지막에 같은 task 사건으로 연결해 reviewer가 재현하게 합니다.

4단계 미니 실습

1. 다섯 관문의 evidence를 한 개씩 찾습니다.
2. critical scenario 12개를 확인합니다.
3. orphan·conflict·regression을 점검합니다.
4. 남은 production gap과 owner를 기록합니다.

16. 합성 actor를 선택하고 학습 경계 확인하기

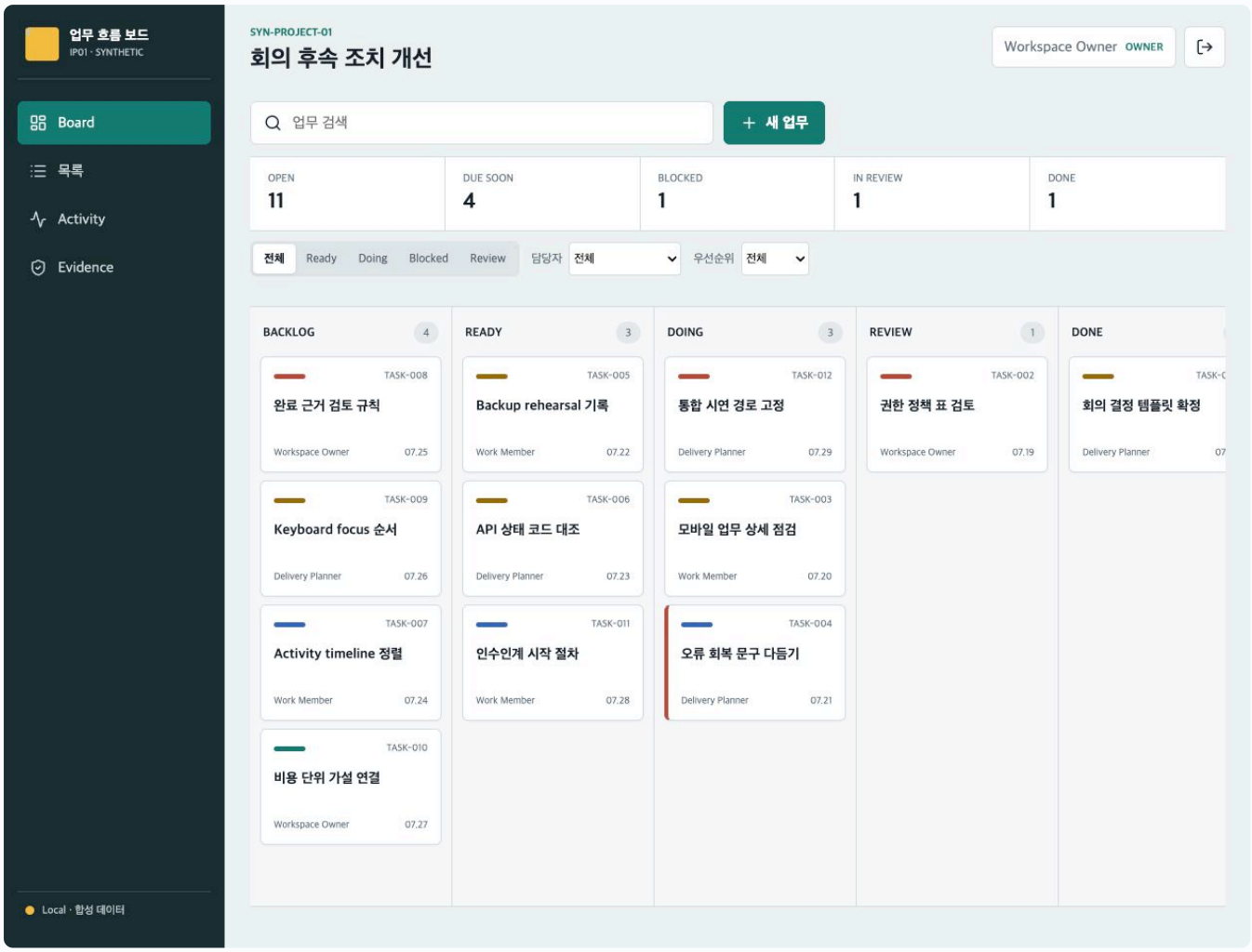


실습 화면 1. 실제 로그인 대신 네 합성 역할을 고르고 개인정보·production 경계를 먼저 확인합니다.

시작 화면은 실제 인증을 흉내 내는 것이 아니라 역할별 권한과 사용자 경험을 비교하기 위한 학습 도구입니다. Owner로 정상 흐름을 수행한 뒤 Viewer로 같은 화면을 열어 변경 명령이 사라지고 서버도 mutation을 거부하는지 비교합니다.

Actor	먼저 해 볼 일	관찰할 차이
Owner	새 업무·편집·상태 전이	모든 command와 evidence
Planner	업무 생성·배정	계획 중심 권한
Member	담당 업무 댓글·진행	resource 조건
Viewer	Board·Evidence 조회	mutation command 0

17. Owner 보드에서 전체 업무 흐름 읽기

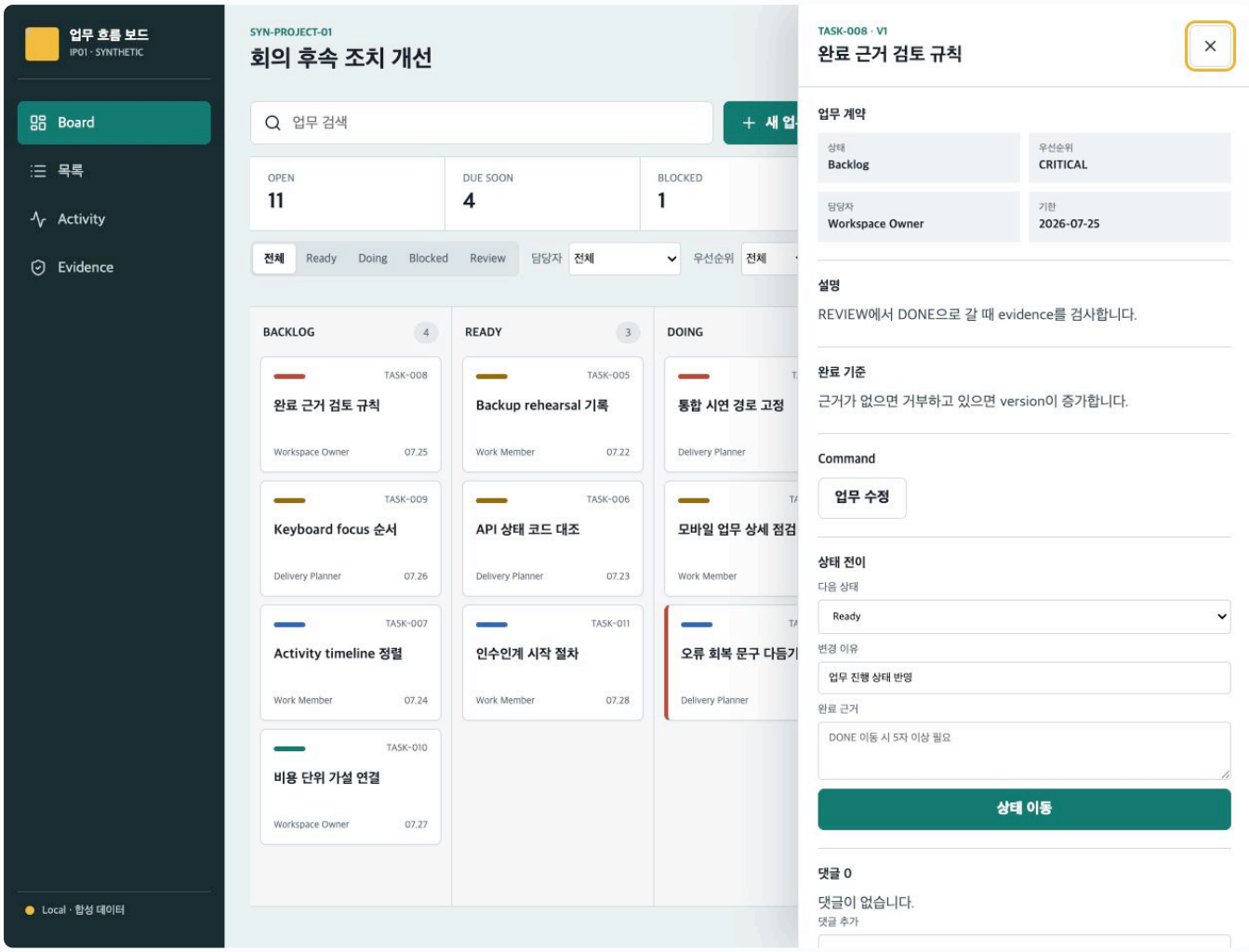


실습 화면 2. 요약·필터·다섯 상태 열을 한 화면에서 훑고 병목을 찾습니다.

먼저 상단 요약에서 Open·Due soon·Blocked·In review·Done을 비교하고, 필터로 상태·담당·우선순위를 좁힙니다. 보드 자체만 가로로 스크롤되도록 해 페이지의 헤더와 command가 흔들리지 않게 했습니다.

읽는 순서	질문	화면 단서
1	마감·차단 위험이 있는가	summary strip
2	어떤 조건으로 좁힐까	segment·assignee·priority
3	어디에 일이 몰렸나	column count
4	어떤 업무를 열까	priority·title·owner·due

18. Task detail에서 계약·명령·이력 연결하기

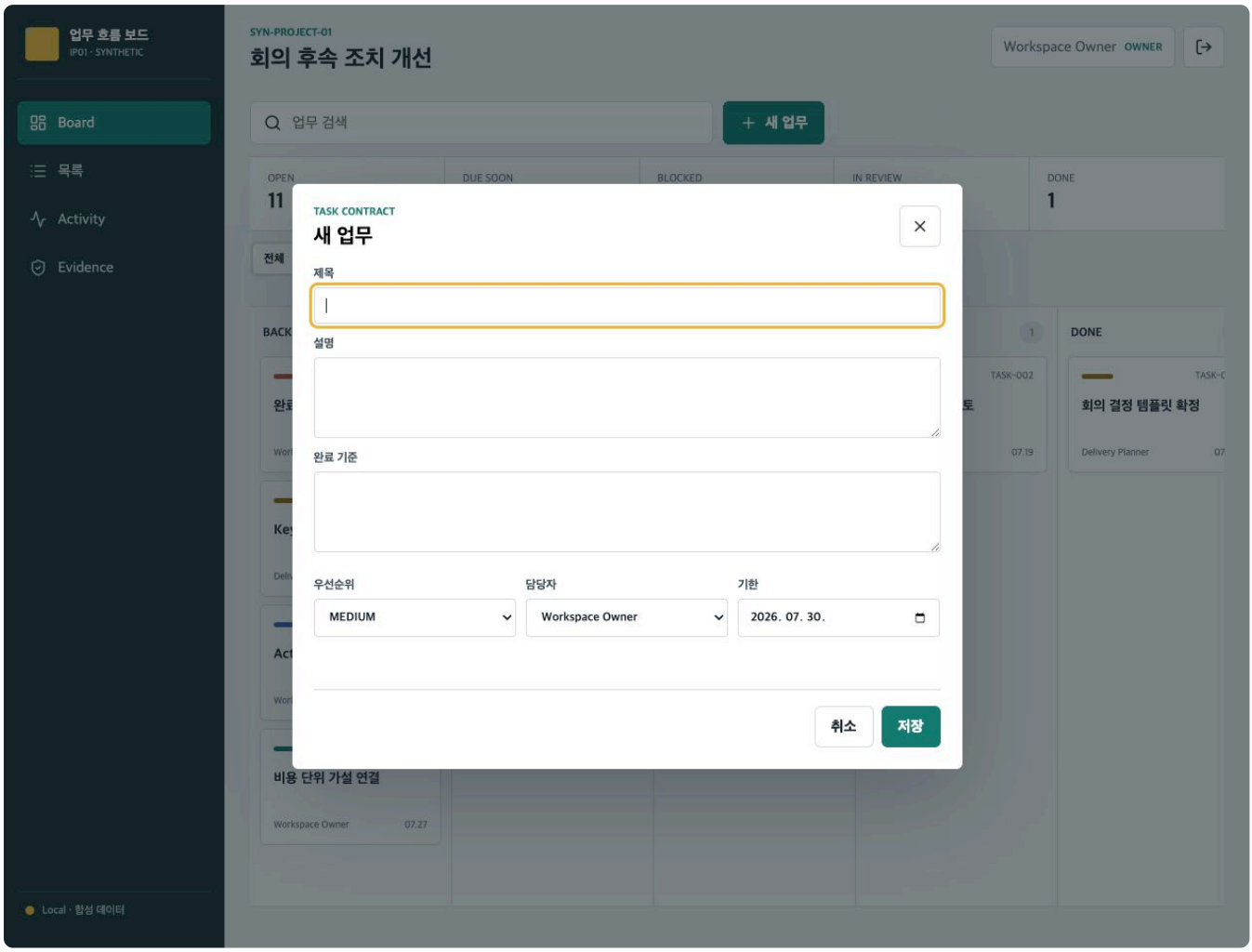


실습 화면 3. 목록 맥락을 유지하면서 업무 계약과 다음 명령을 같은 drawer에서 처리합니다.

Task detail의 위쪽은 현재 version·상태·우선순위·담당·기한과 설명·완료 기준을 보여 줍니다. 아래쪽 command는 역할과 현재 상태에 따라 달라지고, transition reason·completion evidence·comment·event history가 같은 task에 쌓입니다.

상세 영역	사용자 질문	서버 계약
Facts	지금 무엇이 사실인가	task representation·ETag
Description/acceptance	무엇을 왜 끝내야 하나	validated fields
Command	내가 무엇을 바꿀 수 있나	authorization
Transition	다음 상태와 이유는	state guard·If-Match
Timeline	무엇이 바뀌었나	event records

19. 새 업무 dialog에서 입력 계약 확인하기

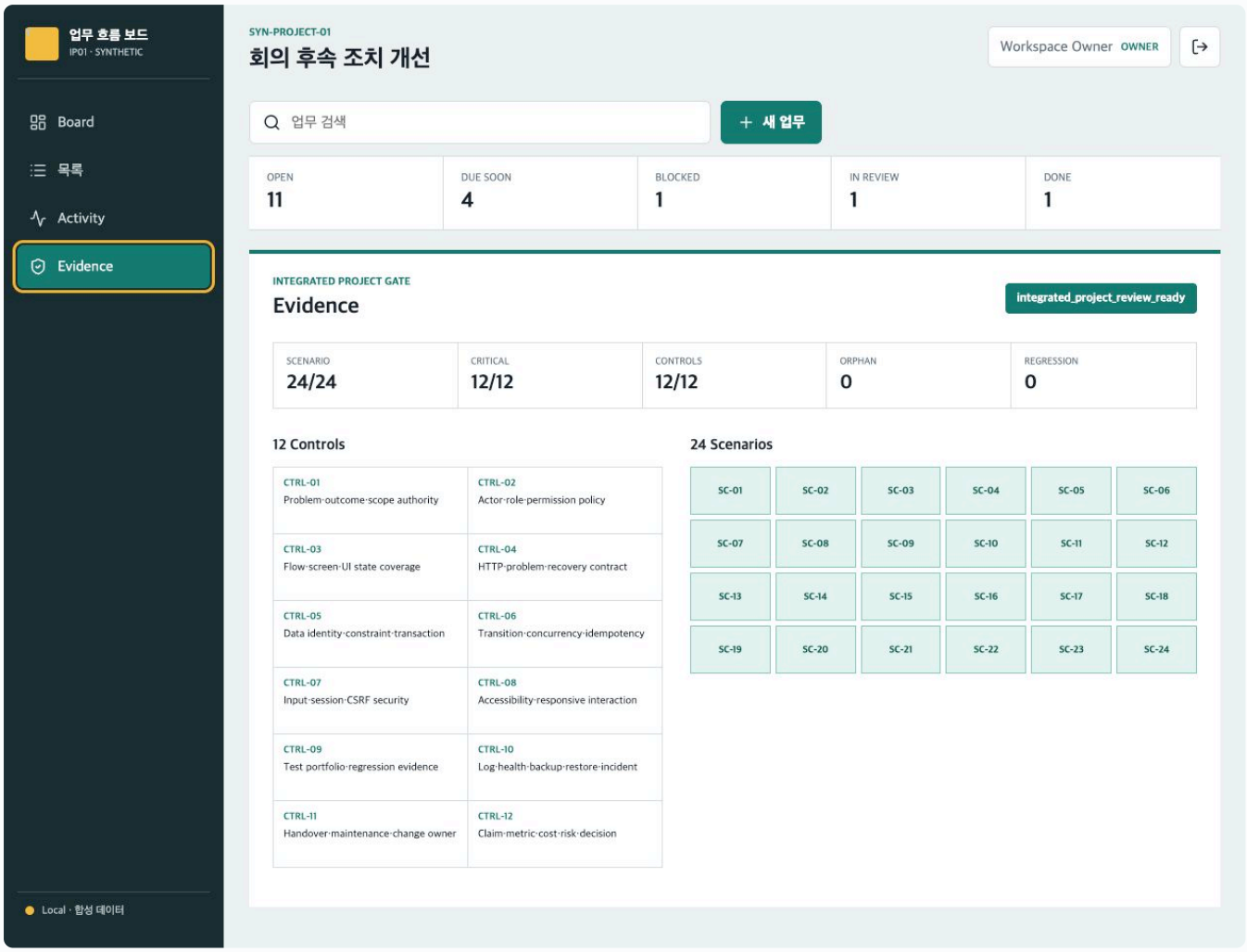


실습 화면 4. 필요한 필드를 한 흐름에 배치하고 validation 오류를 해당 입력 가까이에서 복구합니다.

Dialog는 제목·설명·완료 기준·우선순위·담당·기한을 하나의 task contract로 묶습니다. 저장 중에는 중복 제출을 막고, 같은 Idempotency-Key의 반복 요청은 서버가 첫 결과를 재사용합니다. 입력 오류는 Problem Details의 field 정보와 연결합니다.

입력	왜 필요한가	주요 검사
제목	빠른 식별	필수·길이
설명	맥락과 목적	길이·문자열
완료 기준	Done 판단	구체성·길이
우선순위/담당	정렬·책임	allowlist·actor
기한	시간 경계	날짜 형식·범위

20. Evidence 화면에서 24개 시나리오 읽기

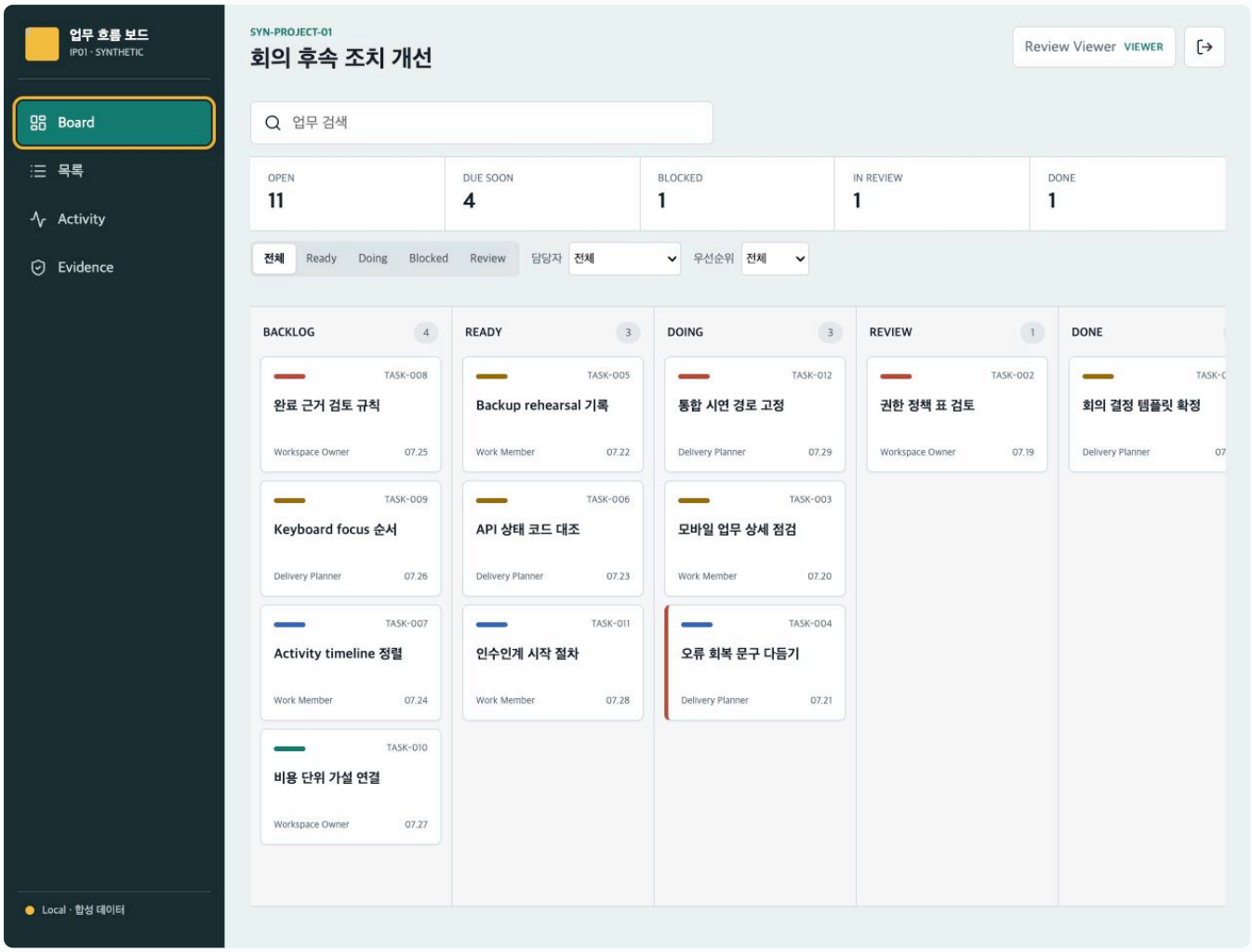


실습 화면 5. Scenario 24/24·Critical 12/12·Controls 12/12·Orphan 0·Regression 0을 함께 봅니다.

Evidence view는 보기 좋은 점수판이 아니라 어떤 위험을 어떤 scenario와 control이 다루는지 찾는 색인입니다. PASS 수를 본 다음 critical scenario, control 이름, 실제 test·audit·trace 파일로 내려가 expected와 actual을 확인합니다.

지표	값	먼저 확인할 질문
Scenario	24/24	모든 acceptance·failure path가 연결됐나
Critical	12/12	실패 시 차단할 항목이 모두 통과했나
Controls	12/12	문제부터 가치까지 통제가 있는가
Orphan	0	연결되지 않은 evidence가 없는가
Regression	0	기존 계약이 깨지지 않았는가

21. Viewer 화면에서 읽기 전용 권한 검증하기

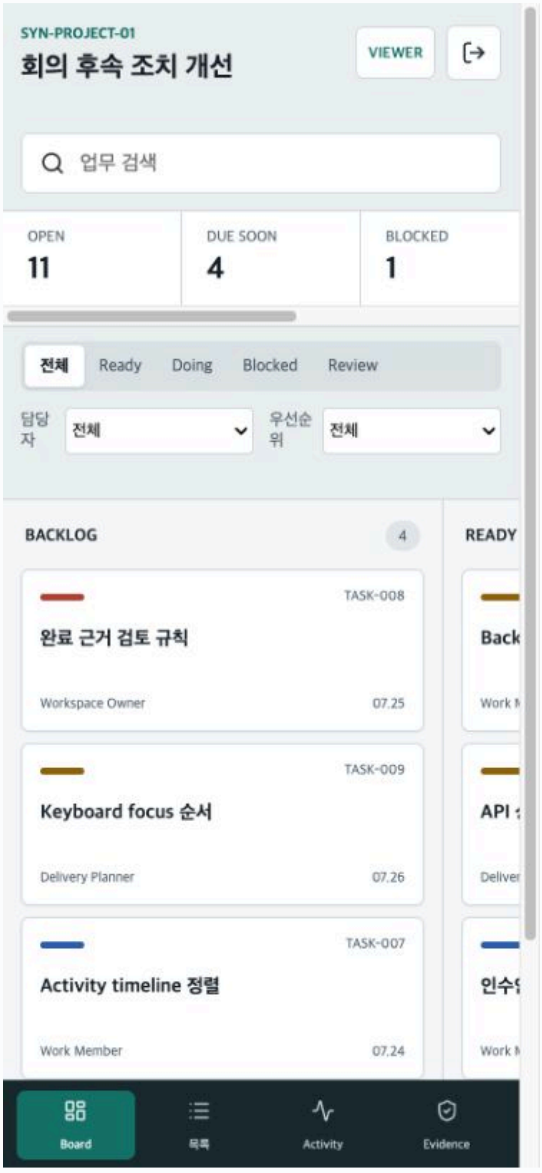


실습 화면 6. 같은 보드 데이터는 보이지만 새 업무 command는 0개입니다.

Viewer 화면은 role label만 바꾼 것이 아닙니다. 생성·수정·댓글·transition command가 렌더링되지 않고, 직접 API mutation을 시도해도 서버가 거부해야 합니다. 허용된 조회와 거부된 변경을 모두 evidence로 남겨야 권한 정책이 설명됩니다.

검사	Expected	Actual 확인
Board 조회	허용	task card 표시
Evidence 조회	허용	24 scenario 표시
새 업무 버튼	없음	button count 0
직접 create API	거부	403 problem
DB task 수	변화 없음	negative test

22. 모바일 보드에서 정보 폭과 조작 안정성 확인하기



실습 화면 7. 390×844에서 요약·필터·보드를 읽고 board만 가로로 탐색합니다.

모바일에서는 sidebar를 하단 navigation으로 바꾸고, actor와 logout 명령을 작게 유지하며, summary와 board를 각각 내부 overflow 영역으로 둡니다. 본문 폭은 100%를 유지해 헤더·필터·하단 navigation이 잘리거나 겹치지 않게 했습니다.

영역	모바일 변화	검사
Navigation	하단 4개 tab	고정 높이·label
Topbar	actor name 축약	title·logout 비겹침
Summary	내부 가로 scroll	page overflow 없음
Filter	두 줄 재배치	label·select 폭

영역	모바일 변화	검사
Board	82vw column	카드 폭·내부 scroll

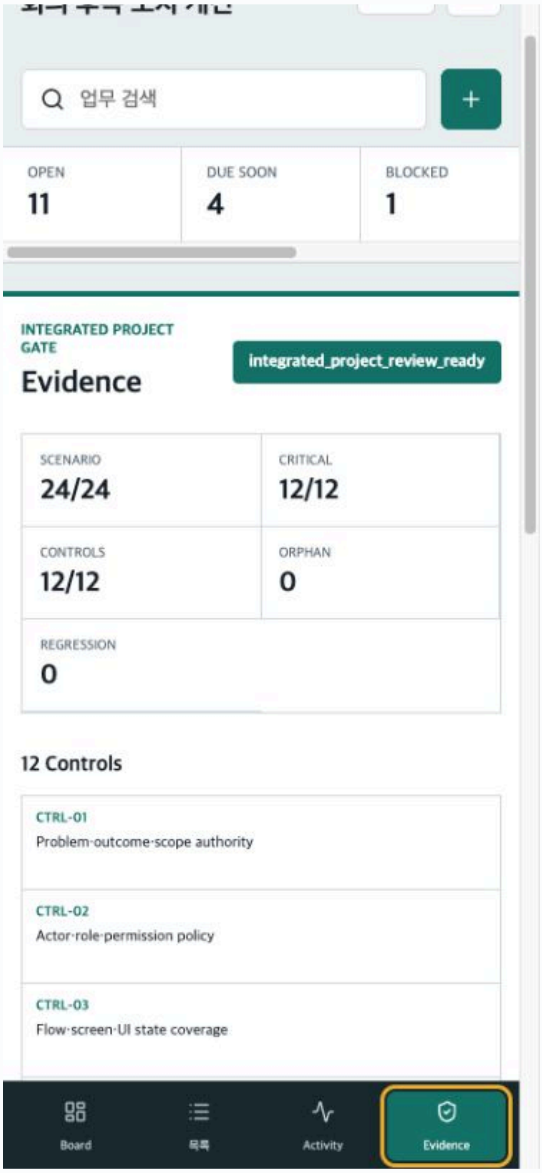
23. 모바일 상세에서 한 손 흐름과 긴 내용 확인하기

실습 화면 8. Drawer가 화면 전체 폭을 사용하고 계약·명령·이력을 세로로 읽습니다.

작은 화면에서는 detail drawer가 100vw를 사용합니다. 닫기 버튼과 heading이 겹치지 않고, facts는 안정된 grid로 보이며, 입력과 command는 손가락으로 누를 수 있는 크기를 유지해야 합니다. 긴 설명과 validation message가 뒤 요소를 밀어내도록 설계합니다.

모바일 상세	확인할 것	실패 예
Header	제목·닫기	텍스트 겹침
Facts	상태·담당·기한	작은 글자·잘림
Command	버튼·select	touch target 부족
Textarea	reason·evidence	가로 overflow
Timeline	긴 내용	아래 요소 가림

24. 모바일 Evidence에서 검증 정보를 재배치하기



실습 화면 9. Gate metric과 control·scenario가 작은 화면에서 한 열 중심으로 재배치됩니다.

Evidence는 정보 밀도가 높아 모바일에서 그대로 축소하면 읽을 수 없습니다. Gate metric을 두 열로, controls를 한 열로, scenarios를 네 열로 재배치하고 페이지 전체 overflow가 아니라 필요한 내부 영역만 흐르게 합니다. 숫자보다 control 이름과 경계 문장이 먼저 읽혀야 합니다.

정보	Desktop	Mobile
Gate metrics	5열	2열 재배치
Evidence grid	2영역	1열
Controls	2열	1열
Scenarios	6열	4열
Navigation	왼쪽 sidebar	아래 fixed nav

25. 세 readiness version과 12×24 evidence 읽기

IP01은 같은 24개 scenario를 세 version에 적용해 '화면 시연', '연결된 서비스', '검증된 통합 서비스'의 차이를 보여 줍니다. 최종 앱은 v3이며 v1·v2는 무엇이 빠지면 판정이 차단되는지 학습하기 위한 비교 모델입니다.

Version	Pass	Evidence	Orphan	Conflict	TBR	Decision
prototype-only-v1	6	4	11	8	9	blocked_demo_without_service_contract
connected-service-v2	15	13	3	2	4	blocked_incomplete_assurance_chain
verified-integrated-service-v3	24	24	0	0	2	integrated_project_review_ready

12 Controls

ID	Control	Minimum evidence
CTRL-01	Problem·outcome·scope authority	prd-and-scope
CTRL-02	Actor·role·permission policy	permission-matrix
CTRL-03	Flow·screen·UI state coverage	screen-state-map
CTRL-04	HTTP·problem·recovery contract	openapi-problem-contract

ID	Control	Minimum evidence
CTRL-05	Data identity·constraint·transaction	schema-and-transaction
CTRL-06	Transition·concurrency·idempotency	state-version-idempotency
CTRL-07	Input·session·CSRF security	security-control-record
CTRL-08	Accessibility·responsive interaction	a11y-responsive-check
CTRL-09	Test portfolio·regression evidence	test-and-regression
CTRL-10	Log·health·backup·restore·incident	operation-evidence
CTRL-11	Handover·maintenance·change owner	handover-package
CTRL-12	Claim·metric·cost·risk·decision	one-page-value-case

24 Scenarios

ID	Scenario	Lane	Control	Risk	Critical
SC-01	problem-outcome link	user-product	CTRL-01	scope	
SC-02	scope and non-goal	user-product	CTRL-02	contract	YES
SC-03	actor job	user-product	CTRL-03	integrity	
SC-04	acceptance trace	user-product	CTRL-04	authorization	
SC-05	metric hypothesis	user-product	CTRL-05	recovery	
SC-06	decision boundary	user-product	CTRL-06	operation	
SC-07	responsive screen	ui-api-data	CTRL-07	scope	
SC-08	UI state recovery	ui-api-data	CTRL-08	contract	
SC-09	HTTP method status	ui-api-data	CTRL-09	integrity	YES
SC-10	problem response	ui-api-data	CTRL-10	authorization	
SC-11	data constraint	ui-api-data	CTRL-11	recovery	YES
SC-12	same-event trace	ui-api-data	CTRL-12	operation	
SC-13	server authorization	permission-failure	CTRL-01	scope	YES

ID	Scenario	Lane	Control	Risk	Critical
SC-14	CSRF and session	permission-failure	CTRL-02	contract	YES
SC-15	validation and output	permission-failure	CTRL-03	integrity	
SC-16	state transition	permission-failure	CTRL-04	authorization	YES
SC-17	version conflict	permission-failure	CTRL-05	recovery	YES
SC-18	idempotent create	permission-failure	CTRL-06	operation	
SC-19	test portfolio	operation-value	CTRL-07	scope	
SC-20	health and audit	operation-value	CTRL-08	contract	YES
SC-21	backup restore	operation-value	CTRL-09	integrity	YES
SC-22	incident rehearsal	operation-value	CTRL-10	authorization	YES
SC-23	handover maintenance	operation-value	CTRL-11	recovery	YES
SC-24	value case handoff	operation-value	CTRL-12	operation	YES

26. 90분 실습 워크북

시간	행동	남길 evidence	통과 질문
0~10분	문제·outcome·scope·boundary	problem/scope note	학습과 production 권한을 나눴나
10~20분	actor·permission	role matrix	UI와 서버 정책이 모두 있나
20~30분	view·state	screen/state map	핵심 일을 끝낼 수 있나
30~40분	UI·API·DB same-event	trace record	task ID·version이 이어지나
40~50분	session·CSRF·validation	negative evidence	실패 뒤 mutation이 0인가
50~60분	conflict·idempotency	retry record	중복과 충돌을 구분했나
60~70분	test portfolio	307+59+6 result	critical scenario가 연결됐나
70~80분	audit·backup·restore	operation evidence	복원 뒤 무결성을 확인했나

시간	행동	남길 evidence	통과 질문
80~90분	Gate·boundary·handoff	decision record	24/24와 production gap이 함께 보이나

```
Problem + user outcome + current alternative + boundary:
Actor + role + allowed/denied action:
View + user question + UI state:
Task state + allowed transition + guard:
UI action + request ID + task ID + ETag/If-Match:
API status/problem + recovery action:
DB row/event/audit + transaction result:
Idempotency duplicate + version conflict evidence:
Test layer + scenario/control link:
Backup + restore + integrity/health check:
Gate actual + orphan/conflict/regression/TBR:
Production gap + owner + next experiment:
```

27. 재사용 가능한 Integrated Project Evidence 템플릿

별도 템플릿 `03_Templates/TIP01_integrated-project-evidence-package.md`는 15개 section으로 구성됩니다. 문제·authority부터 actor·화면·API·데이터·보안·시험·운영·가치·Gate·인계까지 프로젝트가 끊기는 지점을 한 파일에서 찾도록 설계했습니다.

템플릿 범위	먼저 채울 최소 항목	빈칸 처리
Problem·actor	job·outcome·role·permission	TBR+owner+due
Flow·contract	screen·state·method·status·problem	critical gap은 BLOCK
Data·safety	identity·constraint·session·CSRF	unknown은 assumption
Test·operation	scenario·trace·restore	실행 evidence 없으면 미통과
Value·Gate	metric·risk·decision·boundary	production 승인 문구 금지

28. 공식 자료와 적용 경계

아래 자료는 API·데이터·접근성·보안·HTTP·안전한 개발 원리를 확인하는 일차 자료입니다. 목록에 있다는 이유만으로 IP01이 표준 적합·보안 인증·production 승인 상태가 되는 것은 아닙니다. 확인 기준일은 2026-07-16이며 실제 적용과 최신판은 권한 있는 담당자가 재확인해야 합니다.

공식 자료	IP01에서 확인할 원리	적용 경계
OpenAPI Specification 3.2.0	path·operation·request·response·schema 계약	명세 작성만으로 구현 일치·보안 보장 아님
JSON Schema 2020-12	JSON 구조·type·constraint 검증	업무 의미·authorization은 별도 규칙
WCAG 2.2	키보드·focus·reflow·name·contrast	자동 검사만으로 전체 접근성 적합 보장 아님
OWASP ASVS 5.0.0	session·access control·validation·logging 검토 질문	체크리스트 사용이 인증·침투시험을 대신하지 않음
RFC 9110 HTTP Semantics	method·status·conditional request 의미	업무별 정책과 recovery는 서비스가 정의
RFC 9457 Problem Details	구조화된 HTTP 오류 표현	내부 비밀 노출 없이 user-safe detail 설계 필요
NIST SP 800-218 SSDF 1.1	안전한 개발 준비·보호·생산·대응 practice	조직 위험·공급망·운영 context에 맞춤 필요
SQLite Transactions	transaction·commit·rollback 의미	다중 서버·대규모 동시성 요구를 해결한다는 뜻 아님
Python http.server	표준 라이브러리 기반 로컬 학습 서버	production 사용 권장 서버가 아님

29. 셀프 테스트 30

1. 통합 프로젝트의 완성 기준은 무엇인가요?

정답 보기

정답: 사용자가 핵심 일을 끝내고 실패에서 회복하는 흐름이 UI·API·데이터·권한·시험·운영 evidence로 끝까지 연결되는 것입니다.

2. integrated_project_review_ready가 뜻하지 않는 것을 세 가지 쓰세요.

정답 보기

정답: 실제 production 적합성, 보안 인증·법률 승인, 고객 효과·사업 성과 보장을 뜻하지 않습니다.

3. IP01의 핵심 사용자 일을 한 문장으로 쓰세요.

정답 보기

정답: 회의 뒤 실행 항목을 담당·기한·완료 기준과 함께 확정하고 상태와 근거를 추적하는 일입니다.

4. 네 합성 actor 역할을 쓰세요.

정답 보기

정답: Owner, Planner, Member, Viewer입니다.

5. Viewer에서 버튼을 숨기는 것만으로 권한 보호가 되지 않는 이유는 무엇인가요?

정답 보기

정답: 클라이언트는 우회할 수 있으므로 서버가 session·role·resource·action을 다시 판정해야 하기 때문입니다.

6. 네 view가 답하는 질문을 요약하세요.

정답 보기

정답: Board는 흐름, List는 비교, Activity는 변경, Evidence는 검증 공백을 보여 줍니다.

7. 여섯 task 상태를 쓰세요.

정답 보기

정답: BACKLOG, READY, DOING, BLOCKED, REVIEW, DONE입니다.

8. DONE 이동에 추가 evidence가 필요한 이유는 무엇인가요?

정답 보기

정답: 상태 문자열만 바꾸는 것이 아니라 완료 기준을 만족했다는 확인 가능한 근거를 남기기 위해서입니다.

9. Same-event trace란 무엇인가요?

정답 보기

정답: 한 사용자 행동을 UI 요청·API 판정·DB 변경·event·audit·시험 결과에서 같은 식별자와 version으로 추적하는 것입니다.

10. 쓰기 요청의 여섯 보호 단계를 쓰세요.

정답 보기

정답: Session→Origin→CSRF→Authorization→Validation→Transaction·Audit 순서입니다.

11. Opaque session token 원문 대신 hash를 저장하는 이유는 무엇인가요?

정답 보기

정답: 저장소가 노출돼도 원문 token이 곧바로 session 탈취에 쓰이는 위험을 줄이기 위해서입니다.

12. HttpOnly cookie가 해결하지 못하는 두 가지를 쓰세요.

정답 보기

정답: 서버 권한 판정과 CSRF·same-origin 검사를 대신하지 못합니다.

13. Problem Details 응답의 핵심 역할은 무엇인가요?

정답 보기

정답: 오류 종류·상태·상세·instance를 구조화해 UI가 올바른 복구 행동을 선택하게 하는 것입니다.

14. 409 version conflict 뒤 올바른 복구 순서는 무엇인가요?

정답 보기
정답: 최신 상태를 reload하고 사용자의 원래 의도를 다시 확인한 뒤 새 version으로 재시도합니다.

15. Idempotency-Key가 막는 문제는 무엇인가요?

정답 보기
정답: 같은 생성 요청이 네트워크·사용자 재시도로 반복돼 중복 자원이 생기는 문제를 막습니다.

16. ETag·If-Match가 막는 문제는 무엇인가요?

정답 보기
정답: 오래된 화면의 수정이 다른 사용자의 최신 변경을 덮어쓰는 lost update를 막습니다.

17. Task와 Event를 분리하는 이유는 무엇인가요?

정답 보기
정답: Task는 현재 상태를, Event는 변경 이력을 책임져 조회와 추적의 목적을 분리하기 위해서입니다.

18. Database transaction이 필요한 예를 하나 쓰세요.

정답 보기
정답: 상태 전이 때 task version 갱신과 event 기록이 함께 성공하거나 함께 취소되어야 합니다.

19. 307개 시험을 portfolio로 나누는 이유는 무엇인가요?

정답 보기
정답: 규칙·계약·DB·서버·화면 위험을 적절한 속도와 위치에서 잡고 실패 원인을 설명하기 위해서입니다.

20. Negative test가 중요한 이유는 무엇인가요?

정답 보기

정답: 정상 동작뿐 아니라 잘못된 권한·입력·상태가 정확히 거부되고 데이터가 바뀌지 않는지 확인하기 위해서입니다.

21. Browser 검증에서 확인한 두 viewport를 쓰세요.

정답 보기

정답: 데스크톱 기본 viewport와 모바일 390×844 viewport를 확인했습니다.

22. 모바일 board에서 페이지 전체 대신 내부 가로 스크롤을 쓰는 이유는 무엇인가요?

정답 보기

정답: 상태 열의 형식을 유지하면서 헤더·필터·하단 navigation의 전체 페이지 폭을 안정적으로 지키기 위해서입니다.

23. Audit에 요청 원문 전체를 남기면 안 되는 이유는 무엇인가요?

정답 보기

정답: Token·민감 입력·불필요한 payload가 장기 기록에 노출될 위험이 있으므로 필요한 요약 필드만 redaction해 남겨야 합니다.

24. Backup이 검증됐다고 말하려면 무엇이 더 필요한가요?

정답 보기

정답: 독립 restore를 실행하고 데이터 무결성·health·핵심 사용자 시나리오를 다시 통과한 evidence가 필요합니다.

25. 세 readiness version의 통과 수를 쓰세요.

정답 보기

정답: prototype-only-v1은 6/24, connected-service-v2는 15/24, verified-integrated-service-v3는 24/24입니다.

26. 최종 Gate의 critical·control·orphan·regression 수치를 쓰세요.

정답 보기
정답: Critical 12/12, controls 12/12, orphan 0, regression 0입니다.

27. Generator가 기존 target을 거부하는 이유는 무엇인가요?

정답 보기
정답: 이전 결과와 새 evidence가 섞여 재현성과 provenance가 흐려지는 것을 막기 위해서입니다.

28. Python http.server 계열 구현의 production 경계는 무엇인가요?

정답 보기
정답: 학습용 로컬 서버이며 실제 production 보안·성능·가용성 요구를 충족하는 배포 서버로 간주하지 않습니다.

29. 통합 package의 다섯 묶음을 쓰세요.

정답 보기
정답: Blueprint, working app, test evidence, learner manual, handoff입니다.

30. 다음 통합 프로젝트 IP02의 주제는 무엇인가요?

정답 보기
정답: AI 문서 검토 서비스이며 IP01의 역할·계약·evidence·운영 구조 위에 AI 입력·출력·평가·안전 경계를 추가합니다.

30. 용어집 300과 다음 통합 프로젝트 IP02

`04_Glossary/GLOSSARY_integrated_work_management_service.md`에는 15개 묶음·300개 unique term이 있습니다. 문제→역할→화면→API→데이터→상태→동시성→보안→오류→접근성→시험→관찰성→복구→구조→Gate 순서로 한 묶음씩 공부합니다.

묶음	핵심 질문
01~03	누구의 어떤 일을 어떤 화면에서 끝내는가
04~07	요청·데이터·상태·version이 어떻게 맞물리는가
08~10	쓰기 요청과 실패·모바일 사용을 어떻게 보호하는가
11~13	어떻게 시험·관찰·복구하는가
14~15	어떻게 재현·검토·인계하는가

다음 통합 프로젝트는 **IP02 · AI 문서 검토 서비스**입니다. IP01에서 만든 actor·권한·HTTP problem·데이터 version·test portfolio·audit·backup·Gate를 재사용하고, 여기에 문서 ingestion·모델 입력/출력 계약·평가 dataset·hallucination/보안 경계·human review·AI 비용·품질 evidence를 추가합니다.

```
IP01 업무관리 서비스
→ actor·permission·view·state·API·DB·test·operations 재사용
→ IP02 document ingestion·chunk·model contract
→ evaluation dataset·grounding·human review
→ AI safety·privacy·cost·quality Gate
```

마지막 확인: 좋은 통합 프로젝트는 기능을 많이 보여 주는 작품이 아닙니다. 한 사용자의 일을 끝까지 연결하고, 잘못된 요청은 안전하게 거부하며, 충돌과 장애에서 복구하고, 다른 사람이 evidence를 따라 같은 결론을 재현할 수 있는 서비스입니다.