치킨게임 — 메인기획서

> 재구성 기획서 (역설계) / 작성 기준 시점: 2026-08-07 (프로젝트 시작 시점) / 문서 버전: Ver 0.1 프로젝트 ID: P010 / 근거 수준: A / 작성: Claude (재구성) 이 문서는 결과물과 제작 기록으로부터 기획 의도를 역추정해 당시 작성한 것처럼 재구성한 것이다. 확인된 사실과 추정을 [근거]/[추정] 태그로 구분한다.

용어. 틱: 게임 내 1분(실시간 500ms). 인벤토리: 플레이어가 소지하는 3칸(작업대 겸용). DOCK: 발주 재료가 도착하는 입고 대기 30칸. 맛 값: 치킨 품질 수치. 요구 맛: 주문별 합격선. 페이즈: 하루 시간대 상태. [근거: 치킨게임_기획서.md §3, §7, §10]

1. 핵심 메카닉스

1.1 시간과 일일 페이즈

시간 진행과 페이즈 상태 머신

- 의도(Why): "하루는 24시간뿐"이라는 압박을 규칙으로 만든다. 대기 시간을 압축하지 않아 새벽의 빈 시간도 비용으로 느끼게 한다.
- 설계(How): 최소 단위 1분, 하루 1440분. 시작은 Day 1 06:00(`START_MIN = 360`). 페이즈는 COMMUTE(05:00 또는 06:00) → PREP(~08:59) → OPEN(09:00~12:00) → LUNCH(12:01~13:00, 스폰 정지) → OPEN2(13:01~21:59) → DINNER(22:00~22:30, 스폰 정지, 미수주 주문 전부 실패) → CLOSING(22:31~) → FORCED(05:00). 시간 점프는 퇴근과 영업정지 두 경우에만 허용한다. 속도는 일시정지/×1/×2/×4. 대안(자동 스킵)은 "항상 늦는다"는 압박을 약화시켜 버렸다.
- 필요(What): `time.totalMinutes`, `phase`, 틱 리듀서, 일일 정산 함수.
- 작업 방법: 절대분→시각 변환 구현(AI) → 페이즈 전이 테스트로 경계(22:30/22:31)를 고정한다.
- 완료 기준: 1440틱이 정확히 하루이며 CLOSING이 22:31에 시작한다.
- 리스크/대안: 구현 중 CLOSING이 22:30에 시작하는 off-by-one이 발견되어 수정했다. 경계값은 테스트로 고정한다.
- 근거: [근거: 치킨게임_기획서.md §3.1–3.3, claude_d88b9294-e9a5-4cb2-ad98-a3457ec5a033.txt]

일시정지 정책

- 의도(Why): 화면을 파악하기도 전에 첫 손님을 놓치는 일을 막고, 정지 상태에서도 모든 조작이 가능해야 한다는 접근성 원칙을 지킨다.
- 설계(How): 타이틀에서 진입하면 06:00에 멈춘 상태로 시작한다(`SET_SPEED 0`). 정지 중 시간이 드는 행동(이동, 손질, 튀기기, 청소 등)을 하면 거부하지 않고 정지를 해제한 뒤 실행한다. 집기/놓기(0분)와 거부된 행동(재료 없는 청소)은 정지를 유지한다. 판정 기준은 "행동이 실제로 작업을 시작했는가"다. 마지막 배속(`lastSpeed`)을 기억해 Space로 복귀한다.
- 필요(What): 스토어 래퍼에서 이벤트 전후 상태를 비교하는 로직.
- 작업 방법: `gameStore.ts`가 리듀서 호출 전 속도가 0이면 시험 실행(probe)해 작업이 시작됐는지 확인한다(AI). 브라우저에서 ×4 정지/복귀를 확인한다.
- 완료 기준: 즉시 행동에서는 정지가 유지되고 시간이 드는 행동에서만 풀린다.

- 리스크/대안: 거부된 행동에서 정지가 풀리면 의도치 않게 시간이 샌다. 이 때문에 위 판정 기준을 택했다.
- 근거: [근거: claude_d88b9294-e9a5-4cb2-ad98-a3457ec5a033.txt, `src/store/gameStore.ts`]

1.2 자원

기력, 생명력, 숙련도의 3축 압박

- 의도(Why): 기력이 처리량의 한계를, 생명력이 게임 오버를 정하게 해 "몸을 태워 시간을 사는" 선택을 만든다.
- 설계(How): 생명력 0~100(0이면 과로사), 기력 0~100(패시브 소모 시간당 1.0). 기력 0이 되면 탈진: 소요시간 ×2, 손질, 튀기기, 청소, 비둘기 포획/공사장 조달 시 생명력 -1, 화면 채도 저하/비네트. 이동, 집기, 식사, 수주, 전달, 발주는 생명력 소모에서 제외한다(0분 행동까지 포함하면 생명력이 즉시 증발). 숙련도는 주문 성공 +0.20, 맛 미달 +0.08, 튀기기/손질 각 +0.02이며 100에서 손질 6→2분, 튀기기 12→6분, 맛 배율 ×1.30, 기력 소모 ×0.70으로 선형 보간한다.
- 필요(What): `hp/stamina/skill` 상태, 행동별 시간/기력 표(`T_*`, `S_*` 상수).
- 작업 방법: 상수 표를 `constants.ts`에 전사(AI), 탈진 범위를 결정 테스트(§17-26)로 고정한다.
- 완료 기준: 탈진 중 이동에는 생명력이 깎이지 않고 튀기기에는 깎인다.
- 리스크/대안: 숙련도의 사용처가 원안에 없어 효과를 새로 정의했다(원안은 "판매량에 따라 증가"만 명시).
- 근거: [근거: 치킨게임_기획서.md §4, §17-4, §17-26, `src/sim/constants.ts`]

1.3 재료, 유통기한, 보관

잔여 신선도(분) 모델

- 의도(Why): 냉장고 출입에 따라 소모 속도가 바뀌어도 일관되게 동작해야 한다.
- 설계(How): 절대 만료 시각이 아니라 인스턴스별 `freshnessLeft`(분)를 매 틱 감산한다. 냉장고에서는 냉장 영향 품목만 감산 속도 ×0.25(기간 4배와 동일). 생닭 240분, 냉동닭 2,880분, 밀가루 1년, 기름 반년, 조미료/위생용품은 무기한. 0 이하가 되면 [상함]이 붙지만 아이템은 사라지지 않고 사용할 수 있다. 상함 재료로 만든 가공품은 `spoiledTaint`를 전파하고, 그 치킨을 팔면 평판 -10이 맛 판정 결과에 중첩된다. 사용 시 확인 다이얼로그를 1회 띄우고 Shift로 건너뛴다.
- 필요(What): `ItemInstance`(instanceId, itemId, qty, freshnessLeft, taste, createdAt, usedItemIds, spoiledTaint).
- 작업 방법: 생닭을 냉장고에 960분 두면 정확히 0이 되는 테스트(AI).
- 완료 기준: 위 테스트 통과, 세션 확인에서 전분 364일/대두유 181일 표시.
- 리스크/대안: 상한 재료를 소멸시키면 "쓸까 말까"라는 핵심 유혹이 사라지므로 소멸시키지 않는다.
- 근거: [근거: 치킨게임_기획서.md §6, §17-9/§17-10, claude_d88b9294-e9a5-4cb2-ad98-a3457ec5a033.txt]

인벤토리 3칸과 닭 비스택

- 의도(Why): 인벤토리 3칸을 게임의 핵심 마찰로 삼는다. 동선과 선행 작업 계획이 판단의 대상이 된다.

- 설계(How): 저장소는 냉장고 100칸(확장당 +50), 선반 40칸(+20), 인벤토리 3칸, DOCK 30칸. 이동은 반드시 인벤토리를 경유한다. 스택은 밀가루, 기름, 조미료, 위생용품만 20개, 같은 신선도 10분 버킷끼리만 합쳐진다. 닭류와 가공품은 1칸 1개. 손질은 닭 1 + 조미료 2가 인벤토리 3칸을 정확히 채우므로 1왕복당 손질 1회가 상한이다.
- 필요(What): `storage.ts`의 insert/merge 로직, 스택 규칙 테스트.
- 작업 방법: 비스택 품목을 칸마다 분할하는 insert 구현 후 시작 재고 45마리가 45칸을 차지함을 확인한다.
- 완료 기준: 시작 냉장고에 냉동닭 45칸이 실제로 그려지고, 4번째 집기가 거부된다.
- 리스크/대안: 구현 직후 `makeInstance`가 비스택 수량을 1로 잘라 시작 재고에 닭이 1마리뿐인 버그가 있었다. 수정 후 회귀 테스트로 고정했다. 닭도 스택을 허용하면 이동 없이 손질 20연속이 가능해 설계 전제가 무너진다.
- 근거: [근거: 치킨게임_기획서.md §7, §17-21, claude_d88b9294-e9a5-4cb2-ad98-a3457ec5a033.txt]

1.4 제작

손질 → 숙성 → 튀기기 파이프라인

- 의도(Why): "미리 준비할 것"과 "지금 만들 것"을 분리해, 선행 작업의 이득과 식는 비용 사이에서 판단하게 한다.
- 설계(How): 손질: 닭 1 + 서로 다른 조미료 2 → 손질된 닭(맛 = 닭 + 조미료 2종 합), 6분/기력 2.0. 60분 뒤 자동으로 숙성된 닭(맛 ×1.5). 숙성 후 240분이 더 지나면 과숙성(×0.8, 1회). 튀기기: 닭류, 손질된, 숙성된 닭 1 + 밀가루 1 + 기름 1 → 치킨, 12분/기력 3.0. 최종 맛 = (닭 맛 + 밀가루 맛) × 기름 배율 × (1 + 0.3 × 숙련/100). 치킨 맛은 `taste × 0.5^(경과분/60)`로 연속 감쇠하고 1.0 미만이면 [식음]. 밀가루를 튀김옷으로 반드시 소모시켜 밀가루 4종의 맛 차이가 의미를 갖게 했다.
- 필요(What): 재료 맛 값 표, 기름 배율(대두 1.00, 해바라기 1.15, 제트항공유 1.60, 구리스 0.45).
- 작업 방법: 4개 빌드(§8.5)의 맛 값을 테스트가 소수 1자리로 재현한다. 숙성 전환(30.0→45.0), 과숙성(45.0→36.0)은 브라우저에서도 확인했다.
- 완료 기준: 생존형 55.0, 표준형 107.5, 고급형 226.4, 범죄형 37.6이 재현된다.
- 리스크/대안: 숙련도는 튀기기 직전 값으로 계산해야 한다(구현 중 테스트가 튀기기 후 값을 읽어 오차가 났다). 튀기기 UI는 실행 전 예상 맛을 보여 준다.
- 근거: [근거: 치킨게임_기획서.md §8, §17-2, claude_d88b9294-e9a5-4cb2-ad98-a3457ec5a033.txt]

1.5 위생도

0~5 게이지와 일일 판정

- 의도(Why): 청소 부담과 처리량을 맞바꾸게 하고, 방치가 폐업으로 이어지는 압력을 만든다.
- 설계(How): 시작 1.0, 높을수록 더러움. 일일 자연 증가 +0.25(확장 레벨당 +0.10), 튀기기 +0.03, 손질 +0.02, 폐기 +0.01, 비둘기 포획 +0.20. 청소 1회(락스/청소용품 소모, 10분, 기력 3.0)는 -0.10. 출근 시 자연 증가를 먼저 적용하고 위에서 순서대로 판정: 5.0 이상 10% 영업정지 30일, 4.0 이상 5% 벌금 2,000,000원, 3.0 이상 3% 클레임(평판 -10), 1.0 이하 2% 입소문(+5). 청소를 하지 않으면 5일째부터 영업정지 구간이다.

- 필요(What): `filth` 상태(내부 소수 2자리, HUD 1자리), 구간 색상 라벨.
- 작업 방법: 청소 0회 타임라인(Day 1~5)을 밸런스 테스트로 검증한다.
- 완료 기준: Day 5에 5.00으로 클램프된다.
- 리스크/대안: 하루 증가분을 상쇄하려면 청소 13회가 필요해 의도적으로 감당 불가 수준이다. `CLEAN_REDUCTION`을 0.15~0.25로 올릴 여지를 남긴다. 원안의 "청소로 위생도 0.1 상승"은 "숫자가 클수록 지저분"과 상충해 -0.10으로 해석했다.
- 근거: [근거: 치킨게임_기획서.md §9, §17-1, §18.4, `src/sim/constants.ts`]

1.6 평판, 손님, 주문

주문 라이프사이클과 맛 판정

- 의도(Why): 평판이 손님 수와 요구 수준을 함께 끌어올려 성장이 곧 난이도 상승이 되게 한다. 요구 맛을 항상 공개해 판단의 핵심을 "무엇을 낼지"에 둔다.
- 설계(How): 출근 시 그날 손님 수를 확정(`round(rep × priceMult × U(0.85,1.15))`, 평판 20이면 약 20명). 스폰 시각은 영업 720분에 가중 분포(11시 1.5, 18~19시 2.0, 20시 1.5)로 샘플링. 주문은 SPAWNED(9분 내 수주) → ACCEPTED(60분 내 전달) → 맛 판정: 현재 맛 ≥ 요구 맛이면 SUCCESS(돈/평판 +0.5), 미달이면 FAILED_TASTE(돈은 받고 평판 -1). 미수주/시간초과는 돈 없이 -1. 요구 맛 = `(15 + rep × 0.85) × U(0.90, 1.10)`. 동시 수주 `3 + 확장 × 2`. 거절은 불가(놓치면 실패). 전달 시 상한/위험 태그/숙련도 처리 후 치킨 소멸.
- 필요(What): 주문 스케줄러, `Order` 구조, 전달 확인 다이얼로그.
- 작업 방법: 같은 시드에서 스케줄이 재현되는지, 9분 미수주가 평판 -1인지 테스트한다(상수 기준).
- 완료 기준: 평판 20에서 하루 17~23명이 스폰된다.
- 리스크/대안: 평판 0 데드락 방지를 위해 "무손님 -1"을 제거했다. 원안 15분 전달은 최속 경로 16분보다 짧아 수학적으로 불가능했다.
- 근거: [근거: 치킨게임_기획서.md §10, §17-22/§17-23, `src/sim/constants.ts`]

1.7 경제

고정비, 집, 확장, 가격

- 의도(Why): 매달 닥치는 고정비가 "계속 돌려야 하는 압박"을 만들고, 집과 확장이 단기 자금과 장기 성장의 교환이 되게 한다.
- 설계(How): 시작 자금 5,000,000원. 월세는 매월(30일 고정) 1일에 `1,000,000 + 500,000×확장 + 500,000×(집 보유)`, 생활비는 10일에 200,000원. 집 포기 시 보증금 +5,000,000원이나 수면 회복(생명력)과 06:00 출근을 잃는다(출근 05:00, 기력 회복 80). 집 상태 변경은 30일 쿨타임. 확장 3레벨(20M/40M/80M)은 저장 칸/수주 슬롯을 늘리고 위생을 악화시킨다. 가격은 -2~+4단계(10,000~40,000원), 1일 1회 1단계. 인상은 평판 -10/손님 ×0.5, 인하는 +5/×2. 배송은 2일 후 DOCK 도착, 500,000원 미만 주문은 배송비 30,000원. 미납 3회 또는 -10,000,000원이면 파산.
- 필요(What): 달력 로직(30일 고정), 발주, 가격, 부동산, 확장 다이얼로그.
- 작업 방법: Day 1/31/61 월세, Day 10/40/70 생활비를 테스트로 고정한다.
- 완료 기준: 첫 화면 시작 시 월세 150만원이 차감된다(브라우저 확인).

- 리스크/대안: 인상 후 인하를 반복하면 평판이 -5씩 순손실이라 가격 조작 악용을 막는다.
- 근거: [근거: 치킨게임_기획서.md § 11, § 17-7/ § 17-8, claude_d88b9294-e9a5-4cb2-ad98-a3457ec5a033.txt]

1.8 퇴근, 수면, 식사

늦은 퇴근의 복리 페널티

- 의도(Why): "늦게 퇴근 → 회복 감소 → 효율 저하 → 더 늦게 퇴근"의 양방향 스노우볼을 구현한다.
- 설계(How): 퇴근은 CLOSING(22:31~)부터. 회복 배율 `= max(0, 1 - 0.10 × 22시 이후 경과 시간)`. 집 보유 시 생명력 +10/기력 +100, 없으면 생명력 0/기력 +80, 기력 회복은 최소 20 보장. 05:00까지 퇴근하지 않으면 강제 정산(배율 0.3, 완벽한 하루 보너스 없음) 후 06:00 출근(0분 점프 방지로 집 유무와 무관). 일 노동 16시간 초과분은 시간당 생명력 -2, 위생 4.0 이상에서 수면 시 -3. 점심(12:01~13:00)/저녁(22:00~22:30) 식사를 거르면 결식(생명력 -5, 기력 -10). 밥은 10,000원/10분으로 생명력 +5, 치킨은 1마리/5분으로 기력 +10.
- 필요(What): 정산 화면(평판 증감 항목별 내역), 회복 계산식.
- 작업 방법: 22:31~22:59 / 23:00 / 00:00 / 02:00 / 05:00의 회복 표를 테스트한다.
- 완료 기준: 강제 퇴근 시 배율 0.3이 적용된다.
- 리스크/대안: 집을 포기하고 매일 02:00에 퇴근하면 일 순감 약 -10으로 10일 내 사망한다. 이는 단기 자금 조달용으로 의도된 비지속 전략이다.
- 근거: [근거: 치킨게임_기획서.md § 12, § 18.5, § 17-26]

1.9 위험 재료와 행정처분

판매 건당 확률 리스크

- 의도(Why): 위험 재료를 쓰지 못하게 막지 않고 "언젠가 걸린다"는 긴장을 만든다.
- 설계(How): 전달마다 사용된 재료 태그별로 독립 판정한다. 인산염 영업정지 0.5%, 마법의 가루 1.2%(원안 5%/10%), 시멘트/구리스 클레임 8%(평판 -15), 비둘기 클레임 12%(-20)와 벌금 3,000,000원 3%.
  영업정지는 30일: 주문 스폰/수입 0, 월세/생활비는 정상 부과, 위생 자연 증가 절반, 발생 즉시 평판 -20.
  정지 중에는 Skip Day로 하루를 통째로 넘길 수 있다(밥 2끼 20,000원 자동 차감, 노동 0). 획득 경로는 계산대 메뉴의 뒷골목(비둘기: 25분, 성공 60%, 1일 4회 시도)과 공사장 야간 조달(23:00~04:59, 40분, 시멘트 3 + 구리스 3, 15% 확률 벌금 500,000원).
- 필요(What): 리스크 태그, 상수 주석(원안값 병기), PRNG.
- 작업 방법: 리스크 확률을 반복 시행으로 실측한다(`balance.spec.ts`).
- 완료 기준: 실측 비율이 상수와 통계적으로 일치한다.
- 리스크/대안: 원안 확률 그대로면 하루 20건에 64%/88%라 사실상 확정 폐업이다. 원안을 쓰고 싶으면 상수 두 개만 바꾸고 양쪽 값을 주석으로 남긴다.
- 근거: [근거: 치킨게임_기획서.md § 13, § 17-3, `src/sim/constants.ts`, `test/balance.spec.ts`]

2. 구현 방법

2.1 클라이언트/데이터 구조

- `src/sim/`: 순수 시뮬레이션. `constants.ts`, `types.ts`, `rng.ts`(mulberry32), `items.ts`, `formulas.ts`, `state.ts`, `daily.ts`(출근 정산/손님 스케줄/퇴근 정산), `tick.ts`, `actions.ts`, `storage.ts`, `index.ts`. [근거: 파일 목록]
- `src/store/gameStore.ts`: Zustand 단일 스토어로 리듀서를 감싼다. 시작 여부(`started`), 마지막 배속, 토스트를 추가로 관리한다. [근거: `gameStore.ts`]
- `src/ui/`: 렌더링 전용. `Stage`(화면), `panels/`(HUD, 주문, 줄, 저장소 그리드, 인벤토리, 토스트), `dialogs/`(전달, 주방, 계산대 메뉴, 정산, 튜토리얼), `art/`(SVG), `hooks/`(게임 클록, 키맵). [근거: 파일 목록]
- `src/save/save.ts`: `localStorage` 직렬화. 접근 실패는 try/catch로 보호. [근거: `save.ts`]
- 빌드: `vite-plugin-singlefile`로 `dist/index.html` 단일 파일을 만들고 `npm run release`가 `../치킨게임.html`로 복사한다. [근거: `package.json`]

2.2 상태 머신과 이벤트

- 모든 변경은 `SimEvent` 유니온(TICK, MOVE, PICKUP, PUTDOWN, PREP_CHICKEN, FRY, CLEAN, ACCEPT_ORDER, SERVE_ORDER, PLACE_SUPPLY_ORDER, SET_PRICE, EAT, CLOCK_OUT, ABANDON_HOUSE, ACQUIRE_HOUSE, EXPAND_SHOP, CATCH_PIGEON, SITE_RAID, DISCARD, DISCARD_ALL_SPOILED, SKIP_DAY, SET_SPEED, CANCEL_BUSY)로 들어간다. 행동은 점유형(`player.busy`)이며 주문 수주만 논블로킹이다. Esc로 진행 중 행동을 취소하면 재료는 반환되지 않는다. [근거: 기획서 §2.3, §4.5]

2.3 핵심 알고리즘

- 손님 스케줄: 가중 누적분포 샘플링 후 중복 분을 1분씩 밀어 유니크화한다. [근거: 기획서 §10.4]
- 맛 감쇠: 저장 값은 생성 시 맛과 `createdAt`뿐이고 현재 맛은 조회 시점에 계산한다(저장 시 상태 정합성 유지). [근거: 기획서 §8.4]
- 손님 외형: `order.id` 해시로 9종 중 결정해 재로드해도 같은 손님이 선다. [근거: 기획서 §14.6.1]
- 애니메이션: `busy.startedAt/durationMinutes`와 현재 분으로 진행률 p를 계산해 CSS 키프레임을 구동한다. 상태를 바꾸지 않아 배속, 정지, 세이브와 정합한다. [근거: 기획서 §14.6.2]

2.4 시스템 간 연결과 도구

- 토스트 소멸은 `clearToast(id)`가 현재 떠 있는 알림이 그 알림일 때만 지우게 해 옛 타이머가 새 알림을 지우지 않게 한다. [근거: claude_d88b9294…]
- 튜토리얼 문구/도해는 `constants.ts` 값을 직접 참조해 수치 변경 시 문구가 어긋나지 않는다. [근거: claude_d88b9294…]
- 도구: Vitest(`npm test`, `npm run balance`), `tsc --noEmit`, Vite dev 서버(포트 5173). [근거: `package.json`, 세션 08-08]

3. 시스템 간 상호작용과 밸런스 기준

변수	영향	기준값	근거
평판	손님 수, 요구 맛	20→20명/요구 28.8~35.2 / 100→100명/요구 90~110	기획서 §10
기력	하루 처리량	가용 약 83.5, 마리당 약 6.2 → 13~14마리에서 탈진	기획서 §18.2
맛 감쇠	실질 마감	반감기 60분. 전달 제한 60분을 채우면 맛이 절반	기획서 §10.5
위생도	처벌 확률	청소 0회 시 Day 5에 5.0 도달	기획서 §18.4
빌드 맛	통과 가능한 평판 상한	생존형 55.0→평판 약 40, 표준형 107.5→약 80, 숙련 100 표준형 139.8→100, 고급형 226.4	기획서 §18.6
집	시간 vs 회복	보유: 06:00 출근/기력 100/hp +10 / 없음: 05:00 출근/기력 80	기획서 §11.3
일 순익	성장 속도	14마리 × 20,000 - 원가 약 190,000원. 30일 고정비 차감 후 약 +3,400,000원. 확장 20,000,000원에는 한참 부족	기획서 §18.1

설계 의도: 하루 평판 순증 약 +2.5로 평판 100까지 약 32일이다. 가격 -1단계 전략은 손님이 2배가 되어도 기력이 못 따라가 실패가 쌓이는 함정이다. [근거: 기획서 §18.3]

난이도 손잡이(조정 순서): 맛 감쇠 반감기 → 청소 효과(`CLEAN_REDUCTION`) → 제한시간. 제한시간은 이미 3배 완화됐다. [근거: 세션 08-08, 기획서 §9.3]

4. 게임의 나머지 부분의 방향

콘텐츠 구조

- 챕터/스테이지 없이 일 단위 무한 루프. 월(30일)이 고정비 주기를 만든다. 엔딩 8종은 달성 후에도 계속 플레이한다: `survive_90`(Day 90), `rep_100`(평판 100.0), `rich`(순자산 1억), `max_expand`(확장 3), `clean_freak`(30일 연속 위생 1.0 이하), `criminal`(비둘기 치킨 100마리), `alchemist`(마법의 가루 200마리/영업정지 0회), `perfect_day`(하루 50건 이상 전건 성공). [근거: 기획서 §19]
- 게임 오버: 생명력 0(`DEATH` "가게는 남았습니다."), 미납 3회/-10,000,000원(`BANKRUPT` "보증금으로도 못 막았습니다."). 결과 화면은 생존 일수, 매출, 성공률, 최고 평판, 영업정지, 벌금 등을 보여 준다. [근거: 기획서 §19.1, §19.3]

메타 진행

- 로그라이크식 영구 성장 없음. 확장/재료 티어/숙련도가 한 판 안의 진행이다. [근거: 기획서 §11.4, §4.4]

UI/UX 방향

- 글로벌: 상단 HUD(Day, 시각, 배속, 돈, 생명력, 기력, 평판, 위생, 숙련, 치킨값), 우측 주문/로그, 하단 인벤토리/진행 바. 상세는 `03_세부기획서_ui-ux.md`. [근거: 기획서 §14.1]

아트/사운드

- 컬러 플랫 SVG, 화면별 지배색. 사운드는 기획 없음. 아트 리소스 계획은 `03_세부기획서_ui-ux.md`에 통합한다. [근거: 기획서 §14.5]

저장/설정

- 슬롯 3 + 퀵(F5/F9) + 오토(매일 출근 시). 타이틀에서 "새로 시작"은 확인 후 자동, 퀵, 슬롯 1~3/튜토리얼 완료 표시를 전부 삭제한다. 옵션 화면(폰트 3단계, 확인 다이얼로그 on/off)은 계획만 있고 구현 미확인. [근거: 기획서 §15.4, §14.4, claude_d88b9294…]

5. 제작 과정 설계

- 순서: 원안 점검(기획) → 사용자 결정 → 기획서 갱신(AI) → 상수/아이템 → 저장소/상태 → 일일, 틱, 액션 → M1 테스트 → 아트 → 스토어/루프 → 키맵 → 패널/다이얼로그 → 정산/튜토리얼 → 브라우저 확인 → 통합 테스트 → 단일 HTML 배포. [근거: claude_d88b9294-e9a5-4cb2-ad98-a3457ec5a033.txt]
- 병렬 가능: 아트(팔레트, 아이콘, 손님)와 sim 코어는 선행 의존이 없어 병렬이 가능하다. 실제는 순차 진행됐다. [추정: 파일 의존성 기준]
- AI/도구 분담: 사용자는 방향과 판단(결정 질문 응답, 난이도/UX 피드백), AI는 명세 갱신, 구현, 테스트, 브라우저 확인. 도구는 Vitest와 개발 서버 브라우저 패널이다.
- 검증 절차: 마일스톤 수용 기준을 테스트로 기록 → 브라우저 실플레이(시간 경과, 손질, 숙성, 튀기기 애니메이션/정지 동작) → 통합 테스트(Day 1 완주, 30일 시뮬) → 문서-코드 정합 확인. 첫 구현에서 버그 1건과 off-by-one 1건이 이 과정에서 발견됐다.
- 인수인계: HANDOFF 없음. 기획서 §17 결정 기록과 변경 이력이 사실상 인수인계 문서다. 상수는 `constants.ts`에서만 바꾼다.

6. 세부기획서 목록

문서	정할 것
`03_세부기획서_ui-ux.md`	화면 4종 구성, 키맵, 가시성 보강(자동 열림, 이름, 카테고리 색), 튜토리얼 10장, 타이틀, 토스트, 아트 리소스 계획
`03_세부기획서_data-balance.md`	아이템 마스터, 상수 표, 주문/경제 수치, 밸런스 검증 시나리오, 난이도 조정 이력, 테스트 매핑