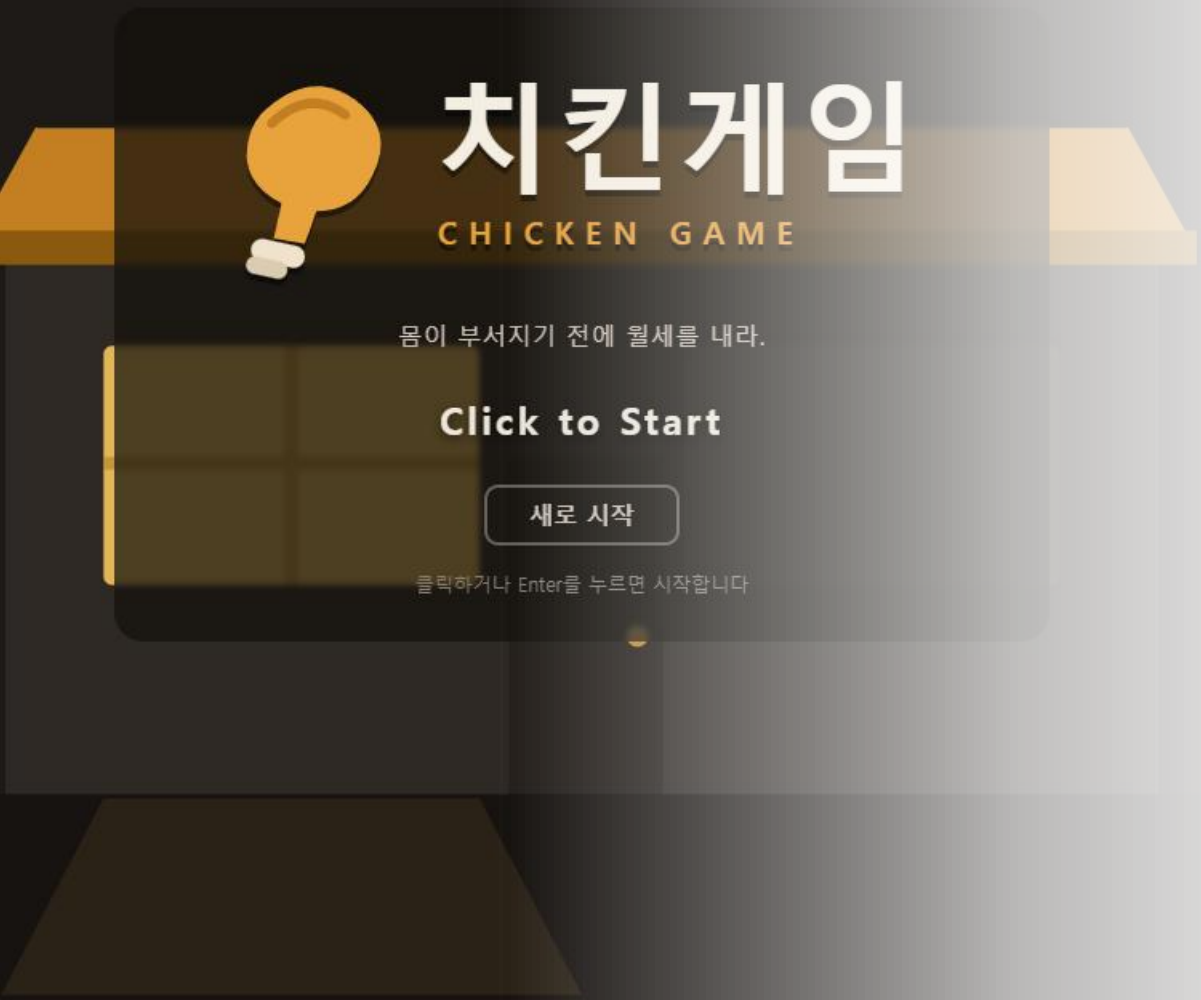


개인 프로젝트

치킨게임
메인 기획서

시간 압박 / 3칸 제약 / 맛 감쇠를 만드는 핵심 메카닉스

문서 최초 작성	문서 최종 수정
20260808	20260808
박도원	박도원

목차

개요 - 2P

1. 목차
2. 로그
3. 문서 계층

현황 - 5P

1. 컨셉 확정 시점의 현 상황

목표 - 6P

1. 컨셉 목적의 재확인

핵심 메카닉스 - 7P

1. 하루 페이즈 루프
2. 치킨 제작 파이프라인
3. 3축 자원 압박
4. 인벤토리 3칸과 닭 비스택
5. 목표 달성을 위한 밸런스 기준

구현 - 12P

1. 구현 구조
2. 기술 선택과 이유
3. 이벤트 처리 흐름
4. 구현에 필요한 것

작업 계획 - 16P

1. 구현 순서와 완료 기준

리소스 - 17P

1. 리소스 수량(핵심 메카닉스 단계)
2. 리소스 규격
3. 생산 방법

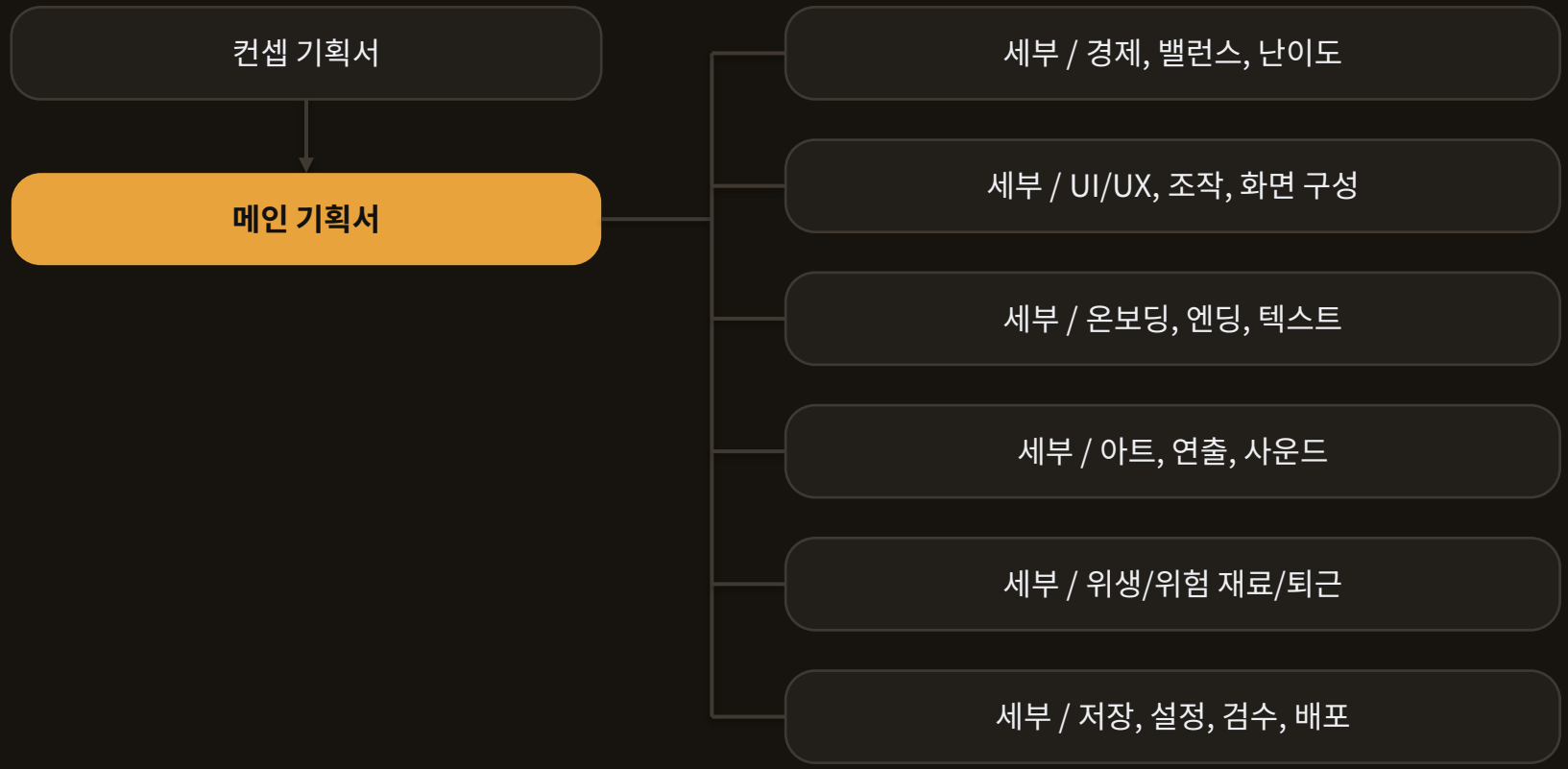
작업 계획 - 20P

1. 구현 리스크

개요 – 로그

순번	작성일	작성자	작성 내용	비고
1	2026-08-08	박도원	스택 확정	TS+Vite+React+Zustand+Vitest, 컬러 플랫 SVG
2	2026-08-08	박도원	기획서 v1.1 갱신	§17.1 결정 19~27, §14.6 신설
3	2026-08-08	박도원	sim 코어 구현	상수, 아이템, 저장소, 일일, 틱, 액션 구현
4	2026-08-08	박도원	M1 테스트 통과	makeInstance 비스택 수량 버그 수정, 48개 통과
5				
6				

개요 – 문서 계층



컨셉 → 메인 → 세부 순으로 상위 문서의 결정을 목표로 받아 구체화함.
강조 표시 = 이 문서

현황 – 컨셉 확정 시점의 현 상황

있는 것

기획서 v1.1과 결정 기록 §17-19~27

의도 확정 규칙을 구현의 단일 근거로 삼으려 함.

근거 사용자 결정이 모두 반영된 상태임.

없는 것

sim 코어, UI, 테스트, 빌드

의도 UI보다 규칙을 먼저 세우려 함.

근거 코드 파일이 아직 없음.

확정된 스택

TypeScript, Vite, React, Zustand, Vitest

의도 AI가 구현하고 테스트로 검증하는 구조임.

근거 기획서 §2.1에 명시됨.

목표 – 컨셉 목적의 재확인



컨셉의 시간 압박/3칸 제약/위험 유혹을 UI 없이 테스트로 증명되는 sim 코어로 먼저 완성하는 것임.

1440틱

정확히 하루가 되는 시간 모델

4빌드

맛 55.0 / 107.5 / 226.4 / 37.6 재현

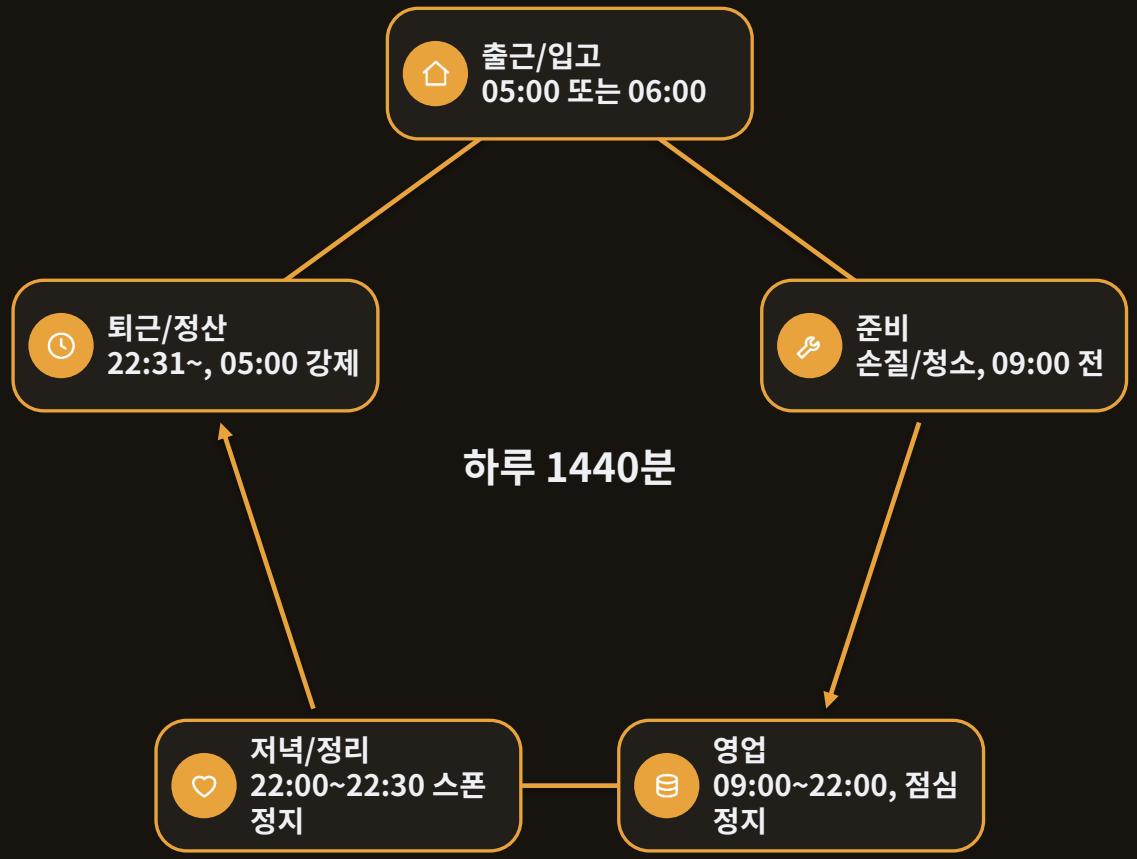
17~23명

평판 20의 Day 1 손님 수

의도 수십 개 규칙을 사람 손 대신 테스트가 검증하게 함.

근거 기획서 §18 밸런스 시나리오가 이미 수치로 정의됨.

핵심 메카닉스 – 하루 페이즈 루프



늦은 퇴근 복리

회복 배율 = 1 - 0.10 × 22시 이후 경과 시간

의도 늦게 퇴근할수록 다음 날 효율이 떨어져 더 늦게 퇴근하게 만듦.

근거 원안 '1시간마다 회복 10% 감소'.

시간 점프 2종만

퇴근과 영업정지에서만 점프 허용

의도 대기 시간도 비용으로 느끼게 함.

근거 자동 스킵은 압박을 약화시킴.

핵심 메카닉스 – 치킨 제작 파이프라인



핵심 메카닉스 – 3축 자원 압박

기력 0~100

처리량의 한계, 하루 약 14마리

의도 몸이 시간을 사는 한계를 만듦.

근거 가용 약 83.5 ÷ 마리당 약 6.2.

생명력 0~100

탈진 시 작업마다 -1, 0이면 사망

의도 자기 착취를 눈에 보이게 함.

근거 원안: 생명력 0이면 사망.

숙련도 0~100

시간/기력은 줄고 맛 배율은 오름

의도 성장 체감을 줌.

근거 원안엔 사용처가 없어 새로 정의함.

핵심 메카닉스 – 인벤토리 3칸과 닭 비스택

규칙

3칸 = 작업대

저장소 간 이동은 반드시 인벤토리를 경유함

닭류 1칸 1개

손질 1회가 닭 1 + 조미료 2로 3칸을 채움

의도

동선이 게임이 됨

1왕복당 손질 1회가 상한이라 선행 계획이 필요함

근거 기획서 §7.3.

스택 허용 시 붕괴

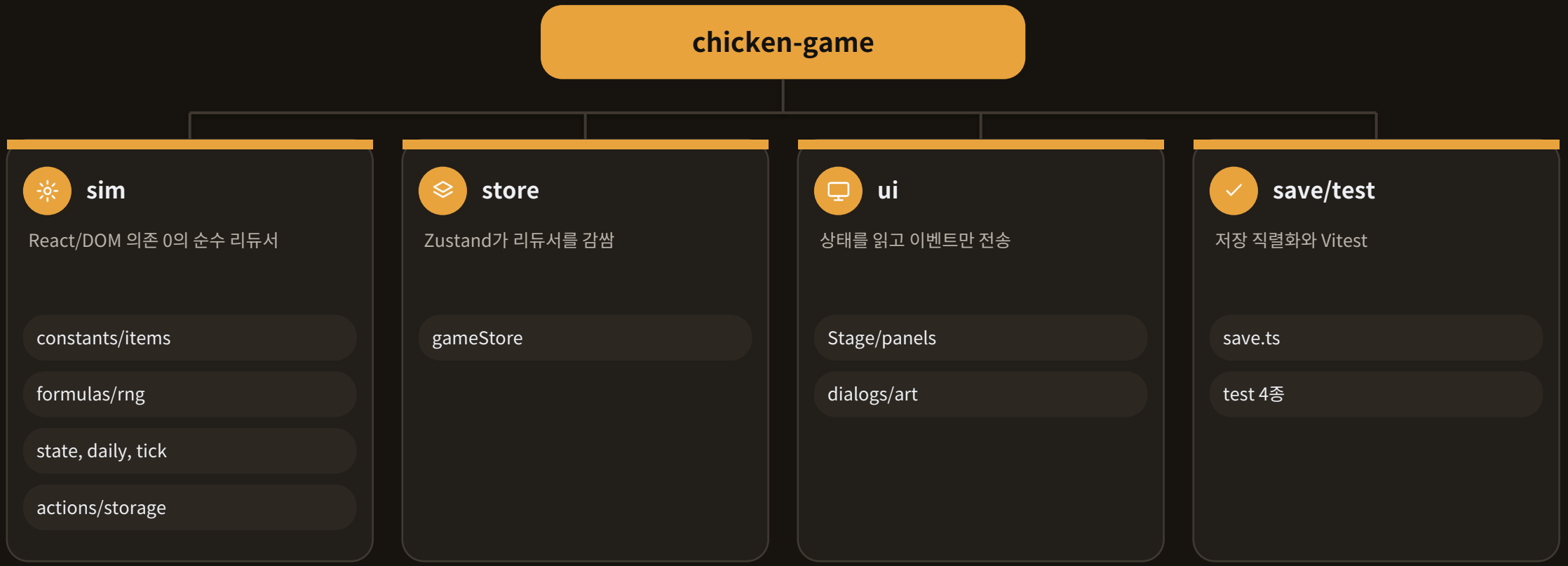
닭이 20개 스택이면 이동 없이 20연속 손질이 가능함

근거 §17-21 사용자 선택.

핵심 메카닉스 – 목표 달성을 위한 밸런스 기준

변수	영향	기준값
평판	손님 수/요구 맛	20 → 20명/28.8~35.2 / 100 → 100명/90~110
기력	하루 처리량	13~14마리에서 탈진
맛 감쇠	실질 마감	반감기 60분, 60분을 채우면 맛 절반
위생도	처분 확률	청소 0회면 Day 5에 5.0 도달
빌드 맛	통과 평판 상한	생존 55.0 → 약 40 / 표준 107.5 → 약 80
일 순익	성장 속도	14마리 기준 약 190,000원, 확장비엔 한참 부족

구현 – 구현 구조


chicken-game
sim
React/DOM 의존 0의 순수 리듀서
constants/items
formulas/rng
state, daily, tick
actions/storage
store
Zustand가 리듀서를 감쌈
gameStore
ui
상태를 읽고 이벤트만 전송
Stage/panels
dialogs/art
save/test
저장 직렬화와 Vitest
save.ts
test 4종

구현 – 기술 선택과 이유

선택	이유	버린 대안
순수 리듀서 reduce(state, event)	같은 시드/입력이면 같은 결과, AI가 테스트로 검증함	컴포넌트 상태에 로직 분산
시드 PRNG(mulberry32), Date/Math.random 미사용	세이브 복원 후에도 난수 순서를 보존함	Math.random
맛은 생성 시 값 + createdAt만 저장	현재 맛은 조회 시 계산해 정합성을 유지함	매 틱 값 갱신
수치는 constants.ts 한 곳	문서, 코드, UI 문구 불일치를 막음	값을 곳곳에 하드코딩
컬러 플랫 SVG, 외부 에셋 0개	단일 HTML 배포가 가능함	이모지/이미지 파일

구현 – 이벤트 처리 흐름


1
입력
키/클릭이 SimEvent로 변환됨
2
리듀서
reduce가 새 상태를 반환함
의도
원본 상태를 변형하지 않음.
3
스토어
Zustand가 상태를 보관함
4
UI 렌더
상태에서 화면/애니메이션이 파생됨
의도
애니메이션이 상태를 바꾸지 않게 함.

구현 – 구현에 필요한 것


핵심 메카닉스 구현
시스템
시간, 자원, 제작, 주문, 경제, 위생
틱 리듀서
일일 정산
행동 점유형 busy
데이터
아이템 21종과 상수 표
ItemDef
시작 재고
검증
마일스톤 수용 기준을 테스트로 고정
sim, 결정, 밸런스
완주 시뮬
도구/인력
기획 1인 + Claude Code
Vitest
Vite 서버

작업 계획 – 구현 순서와 완료 기준

1

상수/아이템

constants.ts, items.ts 전사

의도 수치의 단일 원천을 먼저 세움.

근거 원안 수치표.

2

상태/저장소

types, state, storage

의도 비스택 규칙을 먼저 고정함.

근거 § 17-21.

3

일일, 틱, 액션

daily, tick, actions

의도 페이즈 전이를 코드로 옮김.

근거 § 3/ § 12.

4

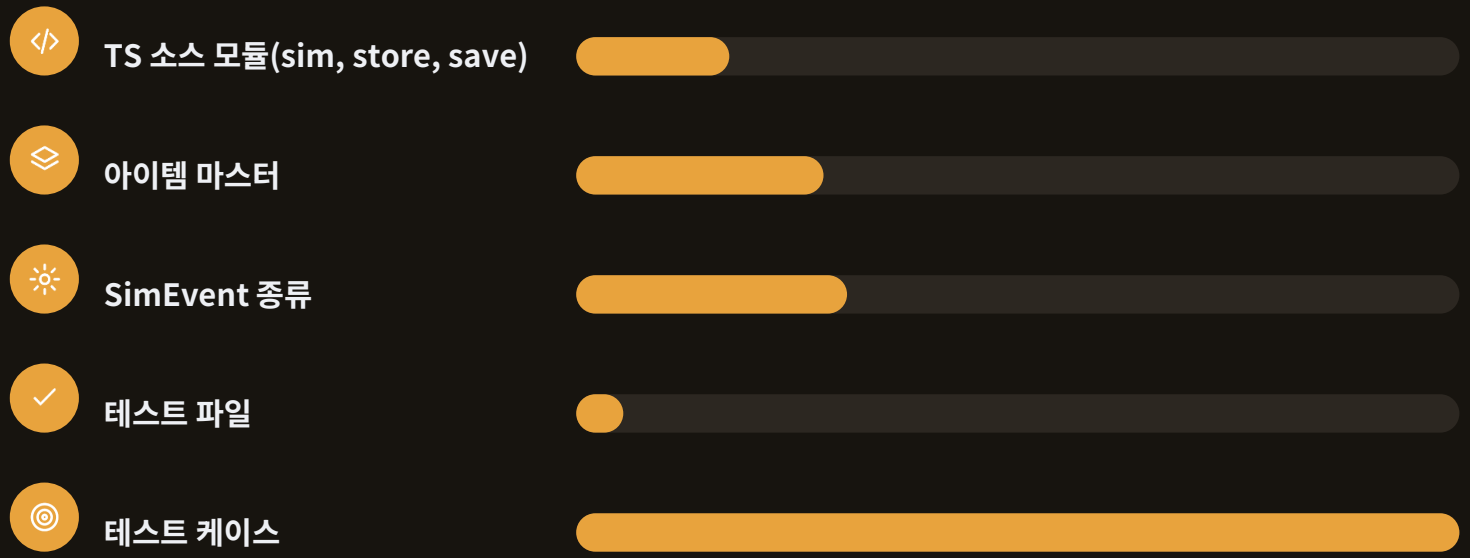
M1 테스트

경계값/4빌드 맛 재현

의도 UI 전에 규칙을 증명함.

근거 완료: 1440틱 = 하루, 4빌드 소수 1자리 재현.

리소스 – 리소스 수량(핵심 메카닉스 단계)



리소스 – 리소스 규격

구분	규격	근거
화면 해상도	최소 1280×720, 스크롤 없음	기획서 §1
그래픽 포맷	인라인 SVG, 팔레트 한 곳(palette.ts)	기획서 §14.5
시간 단위	1분 = 500ms, 하루 1440틱	constants.ts
명명	아이템 id는 snake_case(flour_starch 등), 세이브 키 chickengame.save.<slot>	items.ts, save.ts
배포 파일	단일 HTML 1개, 목표 300KB 이하(최종 실측 약 290KB)	치킨게임.html

리소스 – 생산 방법

1

기획 표

원안 수치를 기획서 표로 고정

2

AI 전사

items.ts/constants.ts로 옮김

Claude Code

3

테스트 검수

빌드별 맛 값이 표와 일치하는지 확인

Vitest

4

단일 빌드

dist/index.html 하나로 묶음

Vite singlefile

작업 계획 – 구현 리스크

⚠ 비스택 품목 수량이 1로 잘려 시작 재고 닭이 1마리뿐일 수 있음	→ 🛡 45칸 점유 테스트를 회귀로 고정
⚠ CLOSING이 22:30에 시작하는 경계 오류가 생길 수 있음	→ 🛡 22:30/22:31 경계 테스트로 고정
⚠ 튀기기 후 숙련도를 읽어 예상 맛이 어긋날 수 있음	→ 🛡 튀기기 직전 값으로 계산하도록 규칙화