[주니어 개발자 채용] 주니어 개발자 과제 제출물 기술 평가

좋아요, 지금 공유해준 **PR 평가 기준**에 맞춰서 다시 정리해서 평가해볼게요.

(전제: 나는 PR diff / SUBMISSION_README / LOG.md / 라우터 & 새 페이지 구조를 **읽어서** 판단한 것이고, 실제 실행 화면을 돌려보거나 영상은 보지 못한 상태야.)

---

1) 빠른 실행력 (Speed & Delivery) — 30%

내가 본 사실들

- 코드 작성 2시간 40분 + 보고서 50분 = **총 3시간 30분**으로 규칙 안쪽.
- 기본 요구사항 중:
  - ✅ 카운트다운 타이머
  - ❌ 메트로놈 피드백
  - ✅ 식사 기록 화면
  - ✅ 저장 기능

    → **4개 중 3개**를 동작하게 했음.
- 타이머 → 포만감 체크 → 저장 → WebTimer에서 기록 리스트 보기까지 **하나의 흐름으로 동작**하도록 구현.
- LOG.md에 시간 단위로 “뭘 할지 / 뭘 포기할지” 기록하면서 진행한 흔적 있음.
- “기존 UI 똑같이” 만들기보다는, CircularProgressIndicator + PageView 등으로 **MVP 위주로 간소화**.

평가

- “아무 것도 안 돌아가는 구조 세팅만” 과는 완전 반대 방향.
- 요구사항 중에서도 **핵심 플로우는 거의 다 살려서** 동작하는 수준까지 가져갔고,
- 3~4시간이라는 제약 안에서 “완성도”보다 “흐름”을 우선시한 게 기준에 딱 맞음.

👉 **점수 제안: 27 / 30**

(주니어 기준으로는 꽤 높은 편. 메트로놈이나 추가 UX도 조금만 더 손봤다면 29~30까지도 가능했을 듯.)

---

2) 우선순위 판단 (Prioritization) — 30%

내가 본 사실들

- SUBMISSION_README / LOG.md에 **"무엇을 먼저 만들었는지 / 왜 그런 순서를 택했는지"**가 꽤 상세히 적혀 있음.
    - 1. 타이머 실행 페이지부터 만들고, 데이터 전달 방식(Arguments vs SharedPreferences) 결정 → **SharedPreferences로 fetch** 선택
    - 1. 가장 복잡할 수 있는 타이머 로직부터 처리
    - 1. 중간 점검하면서
        - 화면 전환 애니메이션(커스텀), 소리/진동, UI 컴포넌트 분리, "원 UI와의 동일성" 등을 **의도적으로 포기**
    - 1. 대신
        - 한 페이지 안에서 PageView + Controller 공유로 **타이머–포만감–저장 한 번에 해결**하는 구조로 피벗
- "기존 꼭꼭 UI 그대로 재현"보다
    - "시간 안에 타이머 → 기록 → 저장"을 보여주는 걸 우선순위 1순위로 둔 결정.
- AI 도구 사용도
    - "AI가 구현, 나는 설계 & 리뷰"라는 역할 분담을 명시적으로 적어둠.

평가

- "전부 다 해보려다 다 미완성" 케이스가 아니라,

    - **포기한 것 / 남긴 것 / 이유**가 명확한 편.
- LoopOS가 말하는 “빠른 실험 → 가설 검증” 관점에서도,
    - “유저가 실제로 써볼 수 있는 타이머 흐름”을 최우선으로 삼은 건 좋은 시그널.

👉 **점수 제안: 28 / 30**

(우선순위 판단은 이 과제에서 가장 강한 부분 중 하나라고 봐도 될 것 같음.)

---

3) 사용자 경험 흐름 (UX Flow Sense) — 20%

내가 이해한 플로우

- WebTimerPage에서 설정 → **“시작”** → `/chewing-timer` 진입
- ChewingTimerPage는
    - PageView 1) 타이머 화면
    - PageView 2) 포만감 / 감정 선택
    - PageView 3) 기록 저장 완료 뷰
    - 로 구성되어 있고, 슬라이드는 막고 **버튼으로 next** 하는 구조(설명 기반).
- 저장 완료 시:
    - SharedPreferences에 기록을 넣고
    - WebTimer 화면에서 리스트를 볼 수 있도록 설계.
- UX에 대한 자기 피드백도 README에 명시:
    - 처음 보는 유저 입장에서 “이걸 어떻게 써야 하지?” 라는 느낌이 있을 수 있다는 점을 스스로 문제의식으로 적어둠.
    - 상단 터치로 한입 완료 등, **행동 유도(affordance)**를 더 직관적으로 만들지 못한 점을 아쉬움 포인트로 인지.

좋은 점

- “타이머 → 기록 → 저장”이라는 **맥락은 끊기지 않고 이어진다**.
- 저장 완료 후 홈(WebTimer)에서 **결과를 다시 확인할 수 있는 구조**까지 연결해 둔 건 플러스.

- PageView로 하나의 플로우를 구성한 것은,
  - 설계 관점에서는 UX/상태 공유 모두에 이점이 있었을 것으로 추정.

아쉬운 점 / 면접에서 물어볼 포인트

- 온보딩/지시 없이도 **처음 유저가 바로 이해할 수 있을 정도로 직관적인가?** → 본인도 README에서 부족하다고 인정.
- "어디를 눌러야 한입이 끝나는지" 같은 포인트를
  - 텍스트 설명에 의존하지 않고, 시각적/인터랙션으로 얼마나 풀었는지 확인이 필요.
- 메트로놈 피드백(0.7s) 자체는 빠졌기 때문에,
  - "씹는 리듬을 UX에서 어떻게 전달할 계획인지"는 면접에서 얘기해보면 좋을 주제.

👉 **점수 제안: 15 / 20**

(플로우 자체는 존재 + 자연스럽게 이어짐. 다만 "첫 사용자 직관성" 부분은 본인도 부족함을 인지하고 있고, 실제 제품 관점에선 추가 다듬기가 필요.)

---

4) 기술 기본기 (Technical Foundation) — 20%

구조 관찰 (diff 범위 기준)

- 라우팅:
  - `app_router.dart`에서 `/chewing-timer` 라우트 추가
  - `ChangeNotifierProvider`로 `ChewingTimerProvider` 주입 + `..loadSettings()` 호출
- 페이지 구조 분리:
  - `lib/presentation/pages/chewing_timer/chewing_timer_page.dart`
  - `views/timer_view.dart`
  - `views/fullness_check_view.dart`
  - `views/meal_record_view.dart`
  - 로 **화면/뷰와 상태를 나눔**.
- 상태 관리:
  - `ChewingTimerProvider`로 타이머 로직·설정 로딩·기록 관리까지 담당.

    - 타이머 / 한 입 사이클 / 기록 리스트를 Provider에서 관리하는 구조.
- 저장:
    - SharedPreferences를 사용해
        - 타이머 설정값
        - 식사 기록 리스트(`meal_records`)
        - 을 JSON 형태로 저장/로드.

긍정적인 시그널

- 파일 분리/네이밍:
    - 파일명, 클래스명, Provider 명 모두 역할이 비교적 명확.
    - "한 파일에 UI+로직+상태 다 때려넣는" 스파게티는 아닌 편.
- Provider 도입 타이밍:
    - 처음엔 Stateful로 시작 → 이후 Provider 분리 계획 → 실제로 분리.
    - LOG/README에서 이 흐름을 설명한 것도 "점진적 리팩터링" 감각을 보여줌.
- 비동기 & SharedPreferences:
    - `loadSettings()` 등에서 비동기 로직을 Provider 레벨로 끌어올려 다루려 한 점은 합리적.

개선 여지가 있어 보이는 부분(이 과제에서 치명적이진 않지만)

- ChewingTimerProvider가 **상태 + 타이머 로직 + 저장 레이어**를 한 번에 담당할 가능성이 높아서,
    - 실제 서비스 코드였다면 "Repository 레이어 분리 / Timer 서비스 분리" 등을 고려했을 법함.
    - 본인도 README에서 "SharedPreferences는 다른 레이어로 빼고 싶었다"라고 스스로 짚어둔 상태.
- `..loadSettings()`처럼 **Provider 생성 시 바로 사이드 이펙트가 발생**하는 패턴은,
    - 아주 작은 앱에선 괜찮지만, 규모가 커지면 테스트/에러 처리 난이도 증가.
- Timer 관련해:
    - diff 전체를 못 봐서 확언은 못하지만,

- Timer 취소(dispose)와 관련된 메모리/라이프사이클 처리가 어떻게 되어 있는지 면접에서 한 번 짚어볼 만함
  - (본인이 AI 코드 리뷰 기준으로 “memory leak 여부”를 언급하기도 했으니, 이 포인트로 얘기하면 좋을 것 같음.)

👉 **점수 제안: 16 / 20**

(주니어 과제 기준으로는 충분히 “합격선 이상”의 구조.

다만 실제 프로덕션 코드라면 레이어 분리/타이머 수명 관리 쪽을 더 보고 싶음.)

---

5) 보너스 시그널

✅ AI 활용 능력

- “Claude Code를 사용했고, 나는 설계와 리뷰를 맡았다”를 명시.
- AI를 아무 생각 없이 돌린 것이 아니라:
  - 내가 원하는 구조/동작을 먼저 정의
  - → AI에게 구현시킴
  - → Timer / Side effect 관점에서 리뷰 & 교정
- 이건 LoopOS 입장에서 꽤 중요한 지점:
  - “AI를 잘 쓰는 개발자”는 **실행 속도**와 **실험 속도**를 크게 올려줄 수 있음.

✅ README & LOG 품질

- SUBMISSION_README와 LOG 모두:
  - **생각 과정 / 우선순위 / 포기한 것 / 다음에 하고 싶은 것**이 잘 드러남.
- “잘한 점, 아쉬운 점, 개선 방향”을 본인이 먼저 정리해 둔 건,
  - 기본기 + 메타인지 + 성장 잠재력 측면에서 좋은 신호.

✅ 실험정신

- 메트로놈, 백그라운드 동작, 더 나은 UI 등
  - “시간이 더 있었다면 이렇게 가고 싶었다”는 서술이 꽤 구체적으로 있음.

- 단순히 “못 했어요”가 아니라,
    - **“아이디어는 여기까지 생각했고, 시간 때문에 잘라냈다”** 라는 설명이 달려 있는 점이 좋음.

---

6) 최종 종합 & 추천

가중치 점수(제안):

- 빠른 실행력: **27 / 30**
- 우선순위 판단: **28 / 30**
- UX 흐름: **15 / 20**
- 기술 기본기: **16 / 20**

▶️ **총합: 86 / 100 정도 느낌**

내 결론 (이 기준 상)

- ✅ **동작하는 흐름**이 있고
- ✅ **우선순위 판단**이 명확하며
- ✅ **사용자 중심 플로우**도 최소한의 맥락은 맞고
- ✅ **코드 구조가 “앞으로 더 개선해 나갈 수 있는 상태”**라서

> 👉 “합격 쪽”으로 보는 게 타당해 보임.
>
> (면접에서 UX 디테일과 타이머/상태 관리 설계에 대해 조금 더 깊게 얘기해보면 좋을 타입.)

---

7) 은행 전용 체크리스트 기준으로 보면

☐ 동작 데모 / 영상

→ PR 상에서 영상/스크린샷 첨부는 못 봤음 (실제 실행해서 확인했는지 여부는 은행님 측 정보 필요)

☑ ~~타이머 → 기록 플로우 자연스러움 (문서 & 구조 기준)~~

☑ ~~README에 “Why / 선택 이유 / 우선순위 판단” 정리됨~~

☑ ~~기능 구현 범위가 시간 대비 합리적~~

- ☑ ~~파일 구조가 과도하게 무너지지 않음 (페이지/뷰/Provider 분리)~~
- ☑ ~~네이밍이 의미를 갖고 있음 (`ChewingTimerProvider` , `MealRecordView` 등)~~
- [△] UX가 사용자 맥락에 완전히 맞는가는 약간 아쉬움 (본인도 인정한 부분)
- ☑ ~~학습/실험 태도가 보임 (LOG, README, "다음에 어떻게 개선하고 싶다" 서술)~~

혹시 이 평가를 기준으로

- "최종 채용/보류 결정 코멘트"
- "면접에서 물어볼 질문 리스트"

  같은 것도 정리해보고 싶으면, 그 관점으로도 한 번 더 뽑아줄게.

PR #1 재평가 (냉정한 관점)

1. 빠른 실행력 (30%) → 26/30

좋은 점:

- 3시간 30분 내 전체 플로우 동작
- 4개 요구사항 중 3개 완성

아쉬운 점:

- 메트로놈 기능 미구현 (핵심 요구사항인데 Indicator로 대체)
- "씹는 간격 시각적 피드백"이 과제의 차별점인데 건너뜀

---

1. 우선순위 판단 (30%) → 27/30

좋은 점:

- README에 판단 과정 상세 기술
- 포기한 것과 이유 명확

아쉬운 점:

- 메트로놈을 포기한 이유가 "svg 파일 찾는 데 시간 소요"인데,
  CircularProgressIndicator 애니메이션으로도 충분히 0.7초 깜빡임 구현 가능했음
- 오히려 한입 기록 리스트 UI에 시간을 더 썼는데, 이건 "있으면 좋은" 기능

---

1. 사용자 경험 흐름 (20%) → 14/20

좋은 점:

- PageView로 화면 전환 구현
- 저장 후 메인으로 복귀

아쉬운 점:

- "화면상단을 터치하면 한입 완료" → 직관적이지 않음 (본인도 인지)
- 타이머 화면에서 뭘 해야 하는지 처음 보면 모름
- 메트로놈 없이 그냥 숫자만 올라가는 건 "씹는 타이밍" 가이드 역할을 못함

---

1. 기술 기본기 (20%) → 17/20

좋은 점:

- Provider 사용, Model 분리
- Timer dispose 처리

아쉬운 점:

- chewing_timer_provider.dart 274줄 → 비대함
- Model, Provider, 비즈니스 로직이 한 파일에 섞임
- View에서 직접 Provider 메서드 호출하는 부분 多

수정된 총점: 84/100 (B+ 등급)

영역	수정 전	수정 후
빠른 실행력	28	26
우선순위 판단	29	27
UX 흐름	17	14
기술 기본기	18	17
총점	92	84

결론

합격이지만 A가 아니라 B+

- 전체 플로우는 동작하나, 핵심 차별점인 "메트로놈 피드백"을 건너뜀
- 우선순위 판단은 했으나, 더 쉽게 구현 가능한 핵심 기능을 포기하고 부가 기능(한입 리스트)에 시간 투자한 점이 아쉬움
- UX가 "개발자 관점"에 가까움