W02
PRD 작성하기

같은 방향을
바라보기 위한 기준

서비스 개요가 담고 있는 것

- 이 서비스를 왜 만들고자 하는가?
- 이 서비스의 핵심은 무엇인가?
- 이 서비스로 해결하고자 하는 문제는 무엇인가?
- 이것을 이해하기 위해 어떤 것들을 알아야 하는가?
- 이를 위해 어떤 것들을 만들고 구현해 내야 하는가?

- 이것을 알고 있다면 이 서비스를 '완벽하게' 만들 수 있을까?

AI가 제일 못하는 것 - 맥락 유지하기

서비스 개요는 '왜 만드는가'에 대한 맥락을 유지하는데 도움이 된다

- 하지만 '어떻게' 만들 것인가의 맥락을 유지하는 데에는 부족하다
- 실제 서비스를 개발하기 위해서 구체적인 결정사항들이 흔들리지 않으려면 무엇이 필요한가?

AI에게 "로그인 기능 만들어줘" → 다음 대화에서 "회원가입도 추가해줘"

- AI는 둘의 연결성(이메일 중복 체크, 비밀번호 정책 일관성 등)을 놓칠 수 있음
- 세션이 길어질수록 초기 결정사항(DB 구조, API 설계 원칙)을 잊어버림
- 여러 AI 도구를 함께 사용할 때 각 도구 간 맥락 공유 불가능

PRD가 해결하는 문제

- AI가 매번 "이전 결정"을 참조할 수 있는 의사결정 지원 문서
- 새로운 세션을 시작해도 일관된 구현 방향 유지
- "왜 이렇게 만들기로 했는지"를 AI에게 지속적으로 상기시킴

PRD

PRD란 무엇인가?

- 제품이 무엇을 해야 하는지(What), 왜 필요한지(Why)를 정리한 문서
- 기술 스택이나 코드 구현 방법(How)은 포함하지 않음
- AI에게 "생각의 가드레일"을 제공하는 역할

AI 에이전트 코딩에서 PRD의 역할

- AI가 코드를 작성할 때 참조할 "판단 기준"
- 구현 중 선택의 순간마다 "이게 맞는 방향인가?" 검증
- 여러 AI 도구 사용 시 일관성 유지 도구

PRD가 없으면?

- AI가 매번 다른 방식으로 구현 (랜덤성)
- 기능 간 충돌 발생 (인증 방식이 페이지마다 다름)
- 개발자가 모든 세부사항을 매번 다시 설명해야 함

기능 정의 X 맥락 정의 O

소프트웨어 개발의 제일 큰 리스트 - 만들었지만 아무도 안 쓰는 것

- 서비스 기획: 이렇게 만들면 서비스를 사용할 것이야
- 하지만 실제로 사용자가 무엇을 필요로 하는지는 출시 전까지 아무도 알지 못한다
- 모든것을 결정하고 만들면 추후 고객에게 적절하지 않은 제품을 만들었을 때 문제를 해결하기 어렵다.

PRD 역시 맥락을 공유하는 문서

- 구체적인 사항을 정해 기록하지 않는다.
- 하지만, 이 제품을 '어떻게'만들어 나갈지에 대해 필요한 정보들을 공유한다.
- 이 문서를 기준으로 AI가 '어떻게' 구현을 할지에 대해 의사결정을 할 수 있다.

1. 배경 및 맥락

이 기능(또는 제품)이 필요한 이유

- 비즈니스 목표: (예: 유지율 15% 향상, 수익 20% 증가)
- 고객 피드백: (예: 사용자 인터뷰, 설문조사 결과)
- 경쟁사 분석: (예: 주요 경쟁사 기능 비교)
- 데이터 인사이트: (예: 신규 사용자 30% 이탈, 결제 전환율 25%)
- 기술적 필요성: (예: 기존 시스템 노후화, API 연동 필요)

2. 목적 및 목표

이 기능이 해결하려는 문제와 목표

- 해결하려는 문제: (예: 결제 과정이 복잡하여 구매 전환율이 낮음)
- 목표:
  - 결제 성공률 (현재: XX% → 목표: YY%)
  - 평균 결제 소요 시간 (현재: XX초 → 목표: YY초)
  - 신규 사용자 첫 결제 완료율 (현재: XX% → 목표: YY%)

3. 주요 사용자 스토리

사용자의 입장에서 기능의 필요성을 설명

- 사용자 유형: (예: 신규 사용자, 기존 사용자, 관리자 등)
- 사용자의 목표: (이 기능을 통해 해결하려는 문제)
- 기대 결과: (사용자가 얻을 수 있는 이점)

예시 형식:

"[사용자 유형]로서, 나는 [목표 또는 행동]을 원한다, 그래서 [얻는 이점]을 할 수 있다."

- 여러개 가능, 꼭 필요한 핵심 행동이 담겨야 한다

4. 성공 지표

이 기능이 성공했는지 판단할 수 있는 기준을 설정

- 정량적 지표 (Quantitative Metrics)
  - 사용자 행동: (예: 기능 사용률 50% → 70%, 평균 사용 시간 3분 → 5분)
  - 비즈니스 성과: (예: 전환율 20% → 30%, 월 활성 사용자 1,000명 → 2,000명)
  - 시스템 성능: (예: API 응답 속도 500ms → 200ms, 에러율 5% → 1%)
- 정성적 지표 (Qualitative Metrics)
  - 사용자 피드백: (예: NPS 점수 40 → 60, 사용자 만족도 설문 3.5점 → 4.5점)
  - 기술적 품질: (예: 코드 리뷰 통과율, 버그 발생 빈도)

AI 에이전트로 개발할 때 활용법

- 성공 지표를 테스트 코드로 변환 요청 가능
- 예: "결제 성공률 95% 이상" → AI에게 "95% 이상 통과하는 테스트 작성해줘"
- 명확한 숫자 목표는 AI가 완료 여부를 스스로 판단하는 기준

5. 제품 개념 및 주요 기능

이 기능이 포함해야 할 주요 요소를 정리

- 핵심 기능
  - 반드시 포함되어야 하는 항목
  - 이 기능의 구현으로 제품의 출시 여부를 결정 할 수 있다
- 이번 릴리즈에서 제외되는 항목
  - 구현의 어려움, 구현 대비 효과가 적어 제외되는 항목
  - AI가 당연히 필요하다고 착각 하는 항목들을 굳이 명시해줘야 혼란이 생기지 않는다

6. UX 및 디자인 가이드

UX/UI 설계에서 고려해야 할 사항을 정리

- 핵심 사용자 흐름: (예: "사용자가 3단계 이내로 결제를 완료할 수 있도록 UI 최적화")
- 디자인 가이드: (예: 주요 UX 원칙, 접근성 고려)

7. 시스템 요구 사항

기술적 요구사항 및 제한 사항을 명확히 작성

- 지원할 플랫폼: (예: Web, iOS, Android)
- 백엔드 요구사항: (예: Node.js + Express API)
- 데이터베이스: (예: PostgreSQL)
- 보안 요구사항: (예: PCI-DSS 준수)

각 기술 요구사항에 대해 이것이 왜 필요한지 확실히 이해해야 한다

- 이해하지 못하고 적용하면 추후 원치 않은 요구사항으로 개발에 어려움을 겪을 수 있다
- AI는 이 기술이 이 기능에 왜 적용되야 하는지 스스로 판단하지 않는다"
  - 명시하지 않으면 AI의 기본 선택(예: React, PostgreSQL)을 따르게 됨
  - 나중에 기술 스택 변경은 처음부터 다시 만드는 것과 같은 비용 발생

8. 가정, 제약 사항 및 의존성

기능 개발 시 고려해야 할 가정, 제약 및 의존성

- 가정 사항 (Assumptions) - 이미 알고있다고 가정하고 진행하는 내용들
  - 예: "대부분의 사용자는 모바일 환경에서 결제를 진행할 것이다."
  - 예: "사용자는 Apple Pay 및 Google Pay를 선호할 것이다."
- 제약 사항 (Constraints) - 미리 제한을 두어야 하는 핵심 요소들
  - 예: "PCI-DSS 보안 규정을 준수해야 하며, 사용자의 카드 정보는 저장할 수 없음."
  - 예: "3rd-party 결제 시스템(Stripe, PayPal)의 API 호출 속도에 의존적임."
- 의존성 (Dependencies) - 다른 중요한 것에 영향을 받을 수 있는 요소들
  - 예: "Stripe 및 PayPal API의 지원 여부"
  - 예: "결제 프로세스 변경에 따른 CS(Customer Support) 대응 매뉴얼 필요"

9. 릴리즈 계획 및 실험 계획

릴리즈 단계: 어떤 단계로 제품을 출시할 것인가

- Alpha 테스트 - 내부 직원 대상 검증(YYYY-MM-DD)
- Beta 테스트 - 기존 사용자 10% 대상 A/B 테스트(YYYY-MM-DD)
- 공식 출시 - 전체 사용자 대상 적용(YYYY-MM-DD)

실험 계획: 제품이 목적을 달성했는지를 검증하는 계획

- A/B 테스트: (예: 기존 결제 UI vs. 개선된 UI)
- 사용자 피드백 분석: (예: 1,000명의 베타 테스터 대상 설문조사)
- 데이터 모니터링: (예: 결제 성공률, 소요 시간 변화 측정)

이것은 예시일 뿐 - 양식은 다양하다

PRD 양식은 프로젝트에 따라 유연하게

- 양식이 아니라 맥락 전달이 목적
- AI가 이해하고 실행할 수 있으면 그것이 좋은 PRD
- 처음엔 간단하게 시작 → 필요할 때 항목 추가
- 필수 항목: 배경 및 맥락, 목적 및 목표, 핵심 기능

실전에서는

- 핵심 항목을 작성하는 것으로 시작
- AI와 대화하며 부족함을 느끼는 부분만 추가
- "AI가 자꾸 잘못 이해하는 부분"을 PRD에 명시
- PRD 작성에 너무 많은 시간 쓰지 말 것 - 코딩하면서 계속 업데이트하는 살아있는 문서

질문 1. 서비스 개요 보완하기

- “서비스 개요를 검토하고, 이 서비스를 내가 의도한 대로 실수없이 만들 수 있도록 하기 위한 PRD를 작성해야 해. PRD를 작성하는데 너가 추가로 더 이해해야 할 정보들이 있는지 검토하고 필요한 것들을 질문해줘"

- 질문에 답을 하면서 AI에게 추가 정보를 제공한다
- 이를 바탕으로 서비스 개요를 더 강화하는 것도 가능
- 확실하지 않은 정보나 부족한 지식이 있다면 AI와 대화하며 학습하고 전달

질문2. PRD 초안 작성하기

- “지금까지 제공한 정보를 바탕으로 MVP를 개발하기 위한 PRD를 작성해줘. 이 PRD를 통해 AI 에이전트가 이 시스템이 필요한 목적을 완전하게 구현할 수 있어야 해"

- 추가 1: “제시하지 않은 내용들에 대해 임의로 가정해서 작성하지는 말아줘"
- 추가 2: “실제 구현시에 더 나은 옵션의 선택을 제한할 수 있는 구체적인 구현을 포함하지 말아줘"
- 추가 3: “실제 코드 내용을 어떻게 작성할지에 대한 내용은 포함하지 말아줘"
- 추가 4: "이 문서를 보는 다른 AI 도구도 동일하게 이해할 수 있도록 작성해줘"

질문 3. PRD 파악하고 개선하기

- 내가 의도한 것과 다르게 적혀 있는 것은 무엇인가?

  -> 피드백 하고 개편하기

- 내가 이해하지 못해서 확인이 어려운 것은 무엇인가?

  -> 해당 내용을 AI와 함께 학습하고 확인 후 검토하기

- 필요하지만 빠져있는, 불필요하지만 포함된 것은 무엇인가?

  -> 필요한 이유, 불필요한 이유를 설명하고 개선 적용하기

질문 4. PRD 최종 검토

- “지금 만들어진 최종안이 처음 의도한 ‘시스템 개요'의 목적 달성에 최적인지 확인하고, 부족하거나 개선할 수 있는 부분이 있는지 검토해줘”

- 검토받은 내용을 바탕으로 보완이 필요한 부분은 추가 보완을 진행
- 그렇지 않은 경우 PRD를 확정하고 실제 구현을 위해 필요한 지식들에 대해 추가 학습을 진행

오늘 해결해야 할 문제

- 작성한 서비스 개요를 바탕으로 PRD 초안을 작성해 봅니다. 초안을 검토해 이 서비스가 어떻게 만들어 지게 될 것인지에 대해 파악합니다.
- 파악한 내용을 바탕으로 추가/개선이 필요한 PRD 항목들에 대해 확인하고 검토 및 개선을 진행합니다. 이 서비스의 개발에 필요한 핵심 내용들이 모두 포함되어 있는지 확인합니다.
- (선택) PRD에 포함될 수 있을 만한 다른 항목들이 있을지 AI와 대화를 통해 제안받고 필요한 부분들이 있다면 나만의 항목을 추가해 의사결정에 더 도움이 되는 PRD로 발전시켜 봅니다.

Q&A