



# Agile환경에서의 AI Lead도전기

AI, 개발을 주도할 수 있을까?

AX Build TF

# AI Lead 목차

## 01 Intro

AI Leading은 AI가 Main, 사람이 Assitant

## 02 상세 분석

AI는 무엇을 잘하고, 무엇을 못하는가?

## 03 결과와 향후계획

성공적인 AI 도입은 '좋은 도구'가 아닌, '좋은 데이터와 명확한 가이드'에 달려있었습니다.

# 1. Intro

AI Leading은 AI가 Main, 사람이 Assistant

# Intro - 우리의 담대한 질문

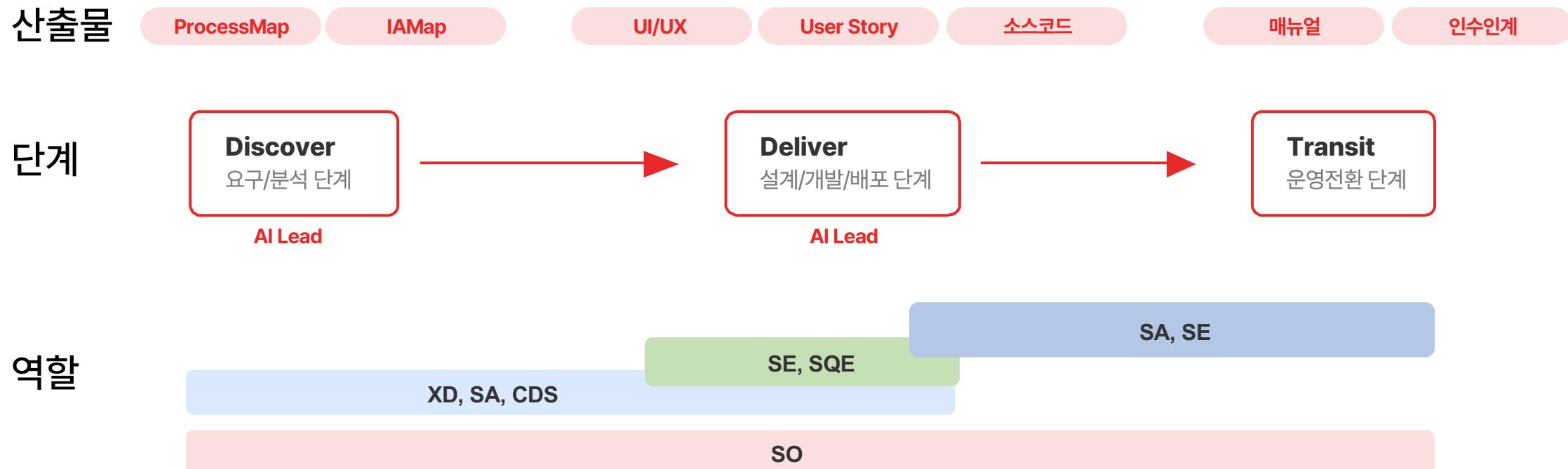
AI가 SW개발 전 과정을 주도하고,  
사람의 개입은 최소화한 채 상용 수준의 품질을 만들 수 있는가?

우리는 질문에 답하기 위해 실제 프로젝트에 AI를 적용하여 검증을 시작했습니다.

- Discover부터 PR까지 전 개발 단계에서 AI가 직접 산출물 생성·검증
- 상용 배포 가능한 수준의 품질을 AI로 확보할 수 있는지 검증
- 실제 프로젝트 환경에서 객관적 측정 지표(AI Leading Score)를 통한 다각적 검증

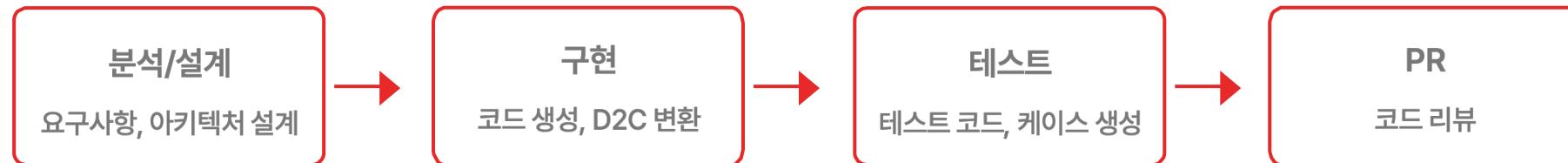
# 왜 Agile에 AI Lead를 접목했나?

2주 단위의 반복적인 Sprint를 통해 점증적 개선과 문서 보다는  
동작하는 SW에 집중하는 실용적인 개발 문화가 AI Lead 도입에 매우 최적화 되어 있음



# AI Lead수행개요

실제 프로젝트 환경에서, 명확한 역할 분담을 통해  
SW개발 단계별로 AI의 실질적 역량을 다각도로 검증



측정 체계

## AI Leading Score (60~100점)

(산출물 품질 × 0.4) + (정확성 × 0.4) + (사용자 만족도 × 0.2)

Excellent	Good	Fair	Poor	Critical
90+	80-89	70-79	60-69	60-



실험 환경

## 3개 POD 운영



POD A: 분석, 설계, 테스트, PR 집중  
POD B: PRD, UI/UX, 프론트엔드 집중  
POD C: 설계, 구현, 테스트 집중



## SW개발 프로젝트 적용

BYBC교육 프로젝트 등 3개 적용  
Sprint 중 복잡도가 낮은 기능 선택

# 어떻게 접근했나?

객관적이고 체계적인 검증을 위해  
AI Leading Score와 AI Rules를 기반으로 한 방법론을 적용



## 실제 프로젝트 적용

3개의 SW개발 프로젝트에 AI를 직접 도입 하여 검증 진행



## AI Leading Score 산출

운영 효율성 + 정확도 + 품질점수를 합산하여 점수 체계화



## AI Rules 품질 관리

프롬프트를 표준화하여 AI 도구·모델이 달라도 일관된 품질 유지



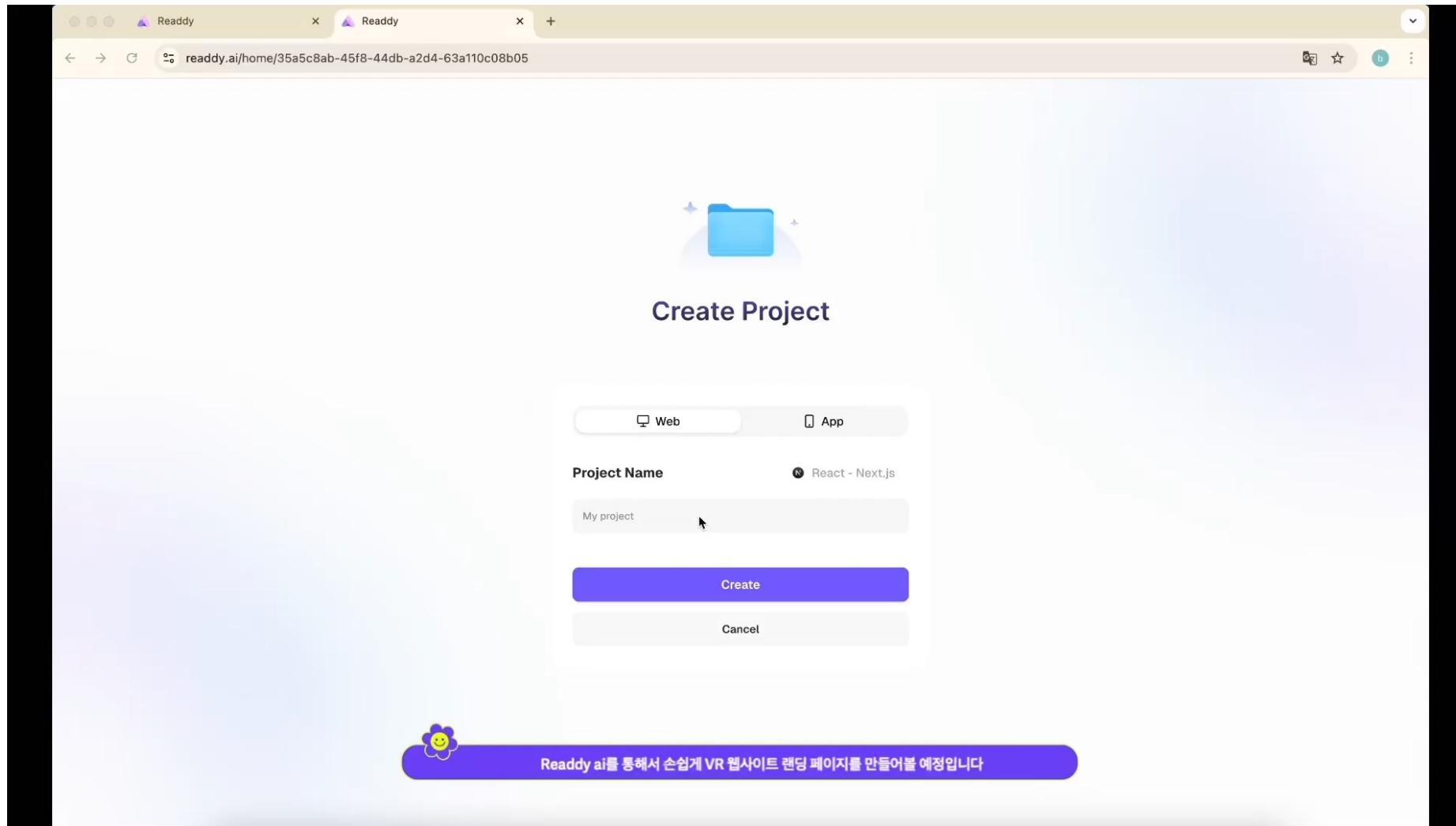
## 피드백 사이클

할루시네이션 사례 DB화 → Rules 재훈련 → 품질 검증  
→ AI Leading Score 향상

## 2. 상세 분석

AI는 무엇을 잘하고, 무엇을 못하는가?

# 1. PRD를 통해 디자인 생성(Readdy.ai)



## 2. 분석/설계 단계

AI가 **기획자와 설계자**의 파트너로서  
주도적으로 역할을 수행 할 수 있는지 확인했습니다.

### Figma → User Story 자동생성

#### 목표

Agile 분석/설계 단계에서 AI가 얼마나 정확하게 유저 스토리를 생성할 수 있는지 검증하고, 플래닝에 도움이 될 자료 제공

#### 대상

고객조회, 결제, 회원가입 등 19개 화면

#### 도구

Claude Sonnet

Figma MCP

Gemini

ChatGPT

### 유저스토리 작성

#### 프롬프트 기본 제공 내용

- 기본 user story 포맷 제공
- 충분한 depth와 범위로 Figma 분석
- INVEST 원칙 + GWT(Given-When-Then) 수용 조건 구체화 명시
- 정보 부족 시 “모름” 출력으로 일정 정도의 추론 허용

#### 프롬프트 최적화

- 유저 스토리 기반 예측 공수/중요도 제공
- 스토리 분할 및 충돌 최소화 제안
- 비기능 요구사항, 테스트 케이스 제안
- Figma 정보 외 Note에 SO가 추가 작성하여 맥락 문제 수동 보완

### 유저스토리 평가

#### 74점

- 기존 SO 생성 스토리의 70%가량 커버, 추가AC도출, 과도한 비기능 요구사항 도출
- Note작성 정도에 따라 스토리포인트 구체화
- 유저 스토리, AC 도출, 스토리 분할 등 반복적/표준화 작업 자동화 유용
- 복잡한 UI/UX 및 Figma 구조 시 유효성 하락

#### 모델별 특성

- (추천) gpt: 자연스러운 스토리텔링
- (추천) claude: 복잡한 figma 분석(긴 컨텍스트, 중첩 json)
- gemini: 다양한 포맷(표, 그림) 이해도 높음

### 3. 분석/설계 단계 - 결과

AI Leading Score

**74점**

실무 적용 가능 수준

시간 단축

**60%** 절감

기준 7MD

AI 적용 3MD

품질 향상

**85%**

INVEST 원칙 준수율

완전 커버:  
50%

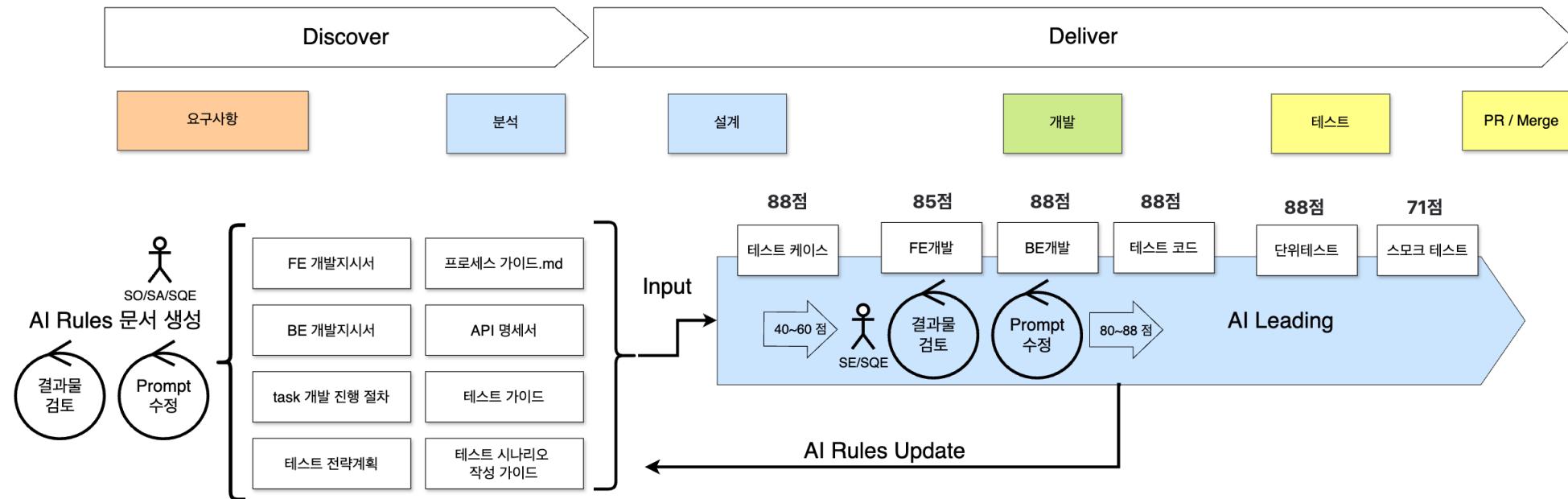
부분 커버:  
36%

미커버:  
14%

구분	상세 내용	영향
성공 요인	구조화된 입력 Figma Description 등 명확한 입력이 품질을 좌우	↑ User Story 품질 ↑ AC 상세도
한계 발견	비즈니스 맥락 부족 과거 히스토리, 전체 서비스 맥락에 대한 이해 부족	↓ 예외 처리 ↓ 일관성
	복잡도 처리 한계 복잡한 업무규칙, 상호 연관 기능 처리 미흡	↑ SO 검증 필요성

## 4. 구현 단계

일관된 AI의 출력 및  
산출물 품질을 높이기 위한 **AI Rules** 작성



### AI Rules란?

AI 기반 개발 도구가 코드 생성, 검토, 분석 시 준수해야 할 일련의 지침이자 규범입니다.  
이를 통해 AI의 출력을 프로젝트 요구사항과 팀 표준에 맞게 일관되게 유지할 수 있습니다.

## 5. 구현 단계

AI가 **개발자의 코딩 파트너**로서  
사람의 개입 없이 코드 생성 수준을 검증

### ▣ Figma → Code 자동 변환 (D2C)

도구:  
Cursor + Claude Code

대상:  
React 컴포넌트, Vue.js 페이지

방법:  
점진적 프롬프트 개선을 통한 정확도 향상

### </> BE/FE 코드 일괄 생성

도구:  
GitHub Copilot + Cursor

대상:  
REST API, 데이터베이스 연동, UI 컴포넌트

방법:  
개발 가이드와 API 명세 기반 자동 생성

### ⟳ 리팩토링 및 최적화

도구:  
Claude Code + CodeRabbit

대상:  
레거시 코드 개선, 성능 최적화

방법:  
코드 분석 후 개선 사항 자동 제안



구현 단계에서 AI는 **코드 작성부터 리팩토링**까지 전 영역에서 활용

# 6. 구현 단계

Copilot 및 Figma MCP를 활용한 FE 개발

Figma 원본



AI 단독 구현(60%수준)



SE 작업(80%수준)



## FE AI Leading 수행 내용

- Figma 작업 결과물을 기본으로 Copilot을 활용하여 FE작업을 진행함

## FE AI Leading 수행 결과

- 현 기준 AI만으로 약 60 % 커버 가능
- 개발자의 프롬프트 추가, 수정 작업을 진행 시 약 80% 까지 커버 가능한 것으로 확인

## 7. 구현 단계 - 결과

| AI Leading Score 평균 83점 (Good) - 프롬프트 템플릿화와 표준화 된 개발에서 AI 품질 향상

### BE 구현

PRD, 개발가이드가 상세할수록 품질 향상

88

Claude Code

Copilot

Taskmaster

### FE 구현

MCP 활용으로 외부정보 접근 용이

85

Copilot

Cursor with MCP

Figma dev mode

### PR 리뷰/생성

코드 검증 품질 우수, 최종검토는 사람 필요

88

Copilot

Claude

CodeRabbit

88

### 테스트 코드

테스트 시나리오 교차검증/커버리지 확보

Cursor AI

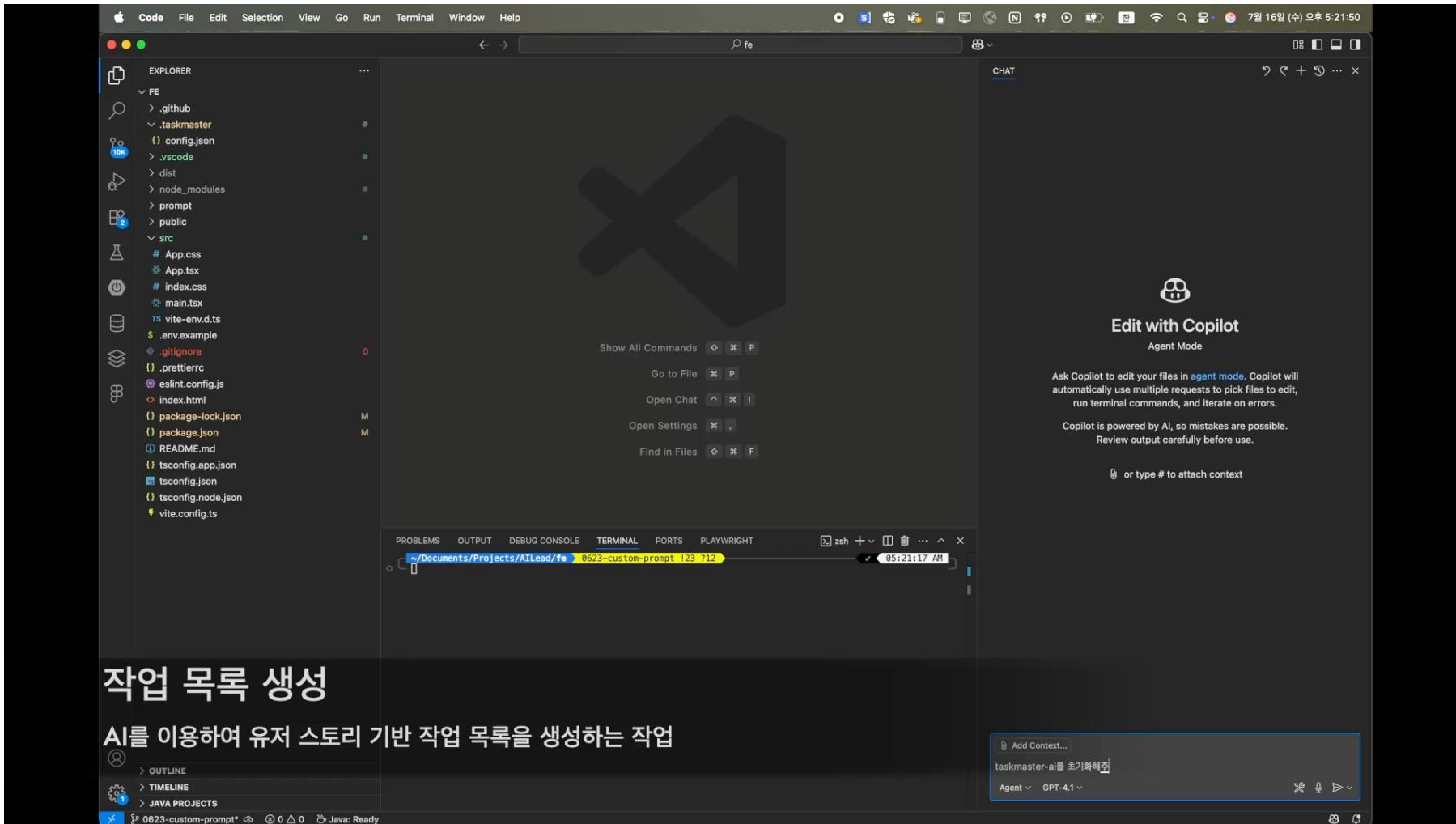
Jetbrain AI

Copilot

### Insight

- 단순/표준화된 작업은 85~90%까지 AI 자동화 가능 - 코딩, 테스트케이스, PR 등
- 복잡한 로직, UI 디테일, 비기능 요구사항, 통합테스트는 SE 보완 필요
- AI Rules, 구조화된 개발 가이드가 AI 출력물 품질의 핵심 요소

## 8. FrontEnd생성(D2C)



## 9. 테스트 단계

AI가 QA의 효율적인 파트너로서 테스트 자동화 검증



### 테스트 케이스 자동생성

- 단위 테스트, 통합 테스트, E2E 테스트 케이스 자동 생성
- AC(Acceptance Criteria) 기반 시나리오 자동화



### 테스트 코드 작성

- Jest, Cypress, Selenium 테스트 코드 생성
- 소스 코드 분석 후 테스트 커버리지 최적화



### 테스트 자동화 스크립트

- Regression 테스트, 성능 테스트 자동화, 과거 테스트 이력 학습을 통한 최적화된 스크립트 생성



### Insight

- 테스트 시나리오, 모듈/유닛 테스트 자동화에서 AI가 가장 빠르게 가시적 성과 획득
- 프롬프트 보완 시 테스트 커버리지는 사람의 보완이 없는 수준까지 달성
- 사용자에 의한 올바른 의존성Mock 설정과 테스트 케이스의 유의미성에 대한 판단 필요
- 또한, 커버리지 분석 도구Jacoco, SnoarQube 등과 연계한 AI 생성 품질에 대한 신뢰 확보 필요

# 10. PR 단계

AI가 **코드 리뷰어 역할 수행이 가능한지 검증**



## PR 템플릿 자동 생성

### 도구

GitHub Actions + CodeRabbit

### 대상

변경사항 요약, 영향도 분석

### 방법

커밋 히스토리 분석을 통한 자동 문서화



## 코드 리뷰 자동화

### 도구

CodeRabbit + SonarQube AI

### 대상

코드 품질, 보안 취약점, 성능 이슈

### 방법

패턴 분석과 Best Practice 기반 리뷰



## 배포 문서 생성

### 도구

Claude + Custom Scripts

### 대상

Release Notes, API 문서, 운영 가이드

### 방법

코드 변경사항 기반 자동 문서 업데이트



리뷰 시간 약 40% 단축



문서화 일관성 향상



자동 보안 취약점 탐지

# 11. PR 단계

활용 영역	AI Leading Score	효율성 개선	주요 특징
 PR 템플릿 자동 생성	73점 (Fair) 	작성 시간 75% 단축 (평균 30분 → 8분)	기본 요약 및 변경 사항 정리 코드 변경 패턴 분석
 코드 리뷰 자동화	88점 (Good) 	리뷰 시간 40% 단축 (평균 2~3일 → 1일 이내)	문법 오류, 보안 이슈, 성능 문제 자동 탐지 리팩토링 제안
 상호작용 기능	85점 (Good) 	커뮤니케이션 효율 60% 향상 (대화형 리뷰 적용 시)	자연어 대화 기능 팀 코드 스타일 학습 적절한 리뷰 어제 안

## 실제 리뷰 사례

### 문법 오류 탐지

- 따옴표 누락: API 응답 처리에서 따옴표 누락 발견
- 메서드명 오타: validateUesr → validateUser 수정 제안
- ✓ 누락된 오류의 90% 이상 자동 탐지

### 코드 품질 개선

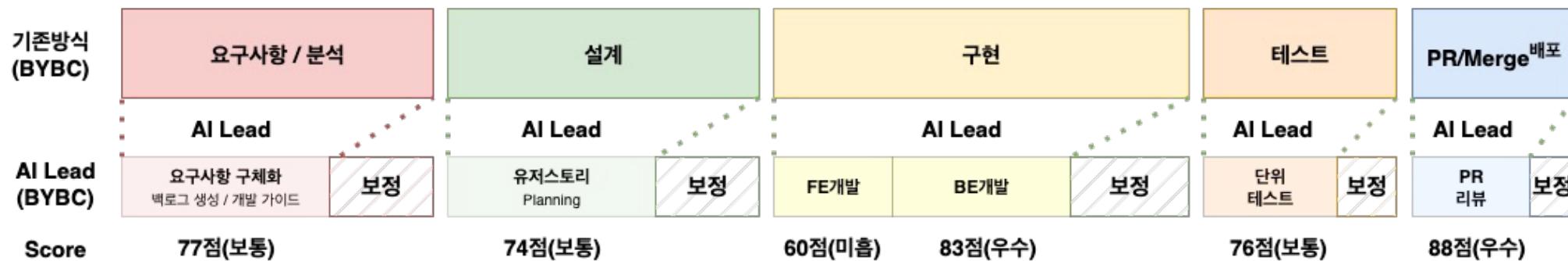
- 테스트 일관성: assertion vs expect 사용 통일 제안
- 중복 코드: 유사 패턴 반복 발견 및 함수 추출 제안
- ✓ 리팩토링 후 코드량 30% 감소

### 3. 결과와 향후계획

성공적인 AI 도입은 '좋은 도구'가 아닌, '좋은 데이터와 명확한 가이드'에 달려있었습니다.

## 12. 실증결과

아직 AI는 개발 전 과정을 이끄는 '주도자'보다,  
개발자의 역량을 강화하는 '조력자' 역할에 적합



※ 각 단계별 수치는 자체적으로 정의한 AI Leading Score(사용자 경험, 재현성, 산출물 품질)를 기준으로 측정  
※ BYBC교육과정 Sprint중 복잡하지 않는 요구사항 기준으로 수행함

### 핵심 인사이트

#### Discover 단계(분석·설계)

초안생성 속도↑, 창의적 설계는 품질 편차커서 사람의 보완필수

#### Deliver 단계(구현·테스트)

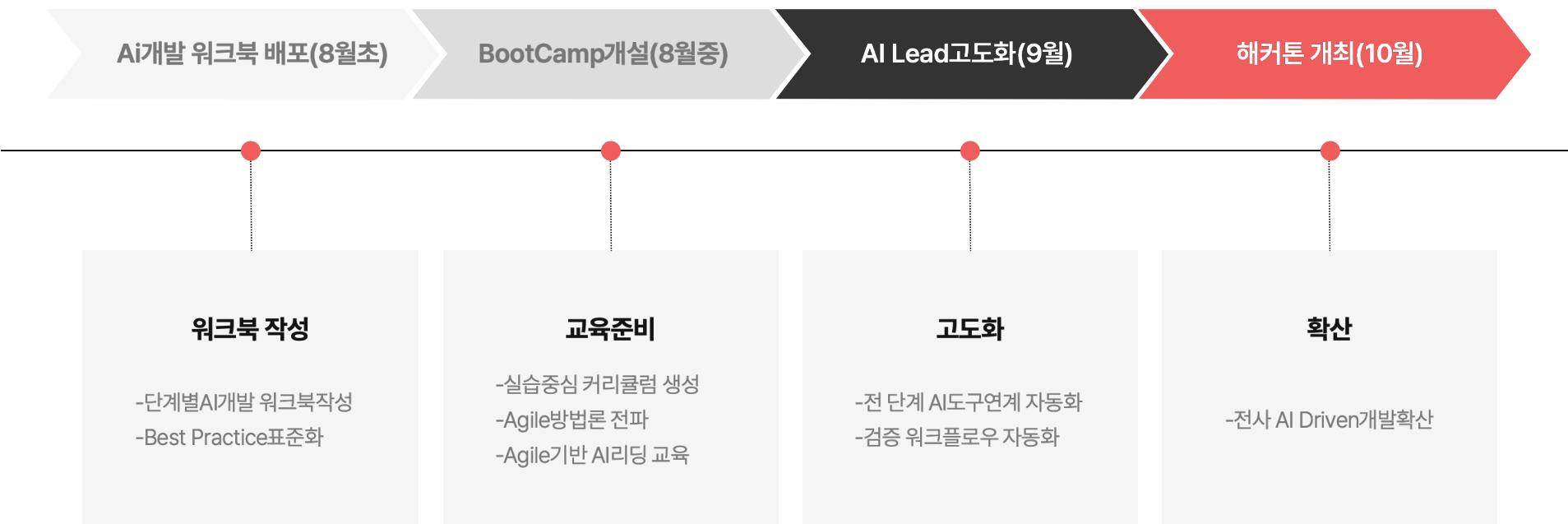
표준화반복적 작업에서 AI 자동화성공, 복잡한 로직예외처리는 한계

#### 보완필요 영역

복잡도 높은 설계/로직, 비기능요구, 코드예외처리, 디자인 정렬 등

# 13. 향후계획

8월 AI 개발 워크북 배포와 부트캠프 개설을 시작으로, 9월 AI Lead 고도화를 거쳐  
10월 해커톤 개최를 통해 AI-Driven 개발 문화를 확산하겠습니다.



# 마무리

AI 주도 개발을 현실로 만들어 나가며,  
AI-Human 협업 모델을 통해 효율적인 개발 문화를 만들어가겠습니다.



AI Leading Score를 통한 객관적 평가 체계 구축



단계별 성과 측정을 통한 실무 적용 가능성 검증



실패 요인 분석을 통한 Best Practice 도출



실무에서 활용 가능한 수준의 AI 역할 재정의

# 본 발표자료도 AI와 함께 만들어졌습니다

훌륭합니다! KT AI Leading PoC 발표 슬라이드가 성공적으로 완성되었습니다.

**완성된 슬라이드 구성**

총 12페이지의 전문적인 발표 자료가 완성되었으며, 다음과 같은 구조로 구성되어 있습니다:

**주요 슬라이드 내용**

- 표지 - "AI, 개발을 주도할 수 있을까?" 핵심 질문 제시
- 목차 - 8개 섹션으로 구성된 발표 흐름
- 오픈닝 - 담대한 질문과 PoC 목적 명시
- 검증 방법론 - AI Leading Score와 체계적 측정 방법
- 평가 기준 - 90점~60점 미만까지 5단계 등급 체계
- 전체 결과 - Discover 77.5점, Deliver 83점 등 핵심 성과
- Discover 분석 - 요구사항·설계 단계 상세 분석 (차트 포함)
- Deliver 분석 - 구현·테스트 단계 상세 분석
- 도구 성과표 - 단계별 Score와 AI 도구 비교 태이블
- 통합 인사이트 - AI Co-pilot 모델 핵심 개념
- 향후 방향 - KT의 AI Co-pilot 고도화 계획
- 마무리/Q&A - 결론과 질문 받기

**디자인 특징**

- KT 브랜드 컬러 (#ED1C24) 활용한 전문적 디자인
- 시각적 요소: 차트, 테이블, 점수 배지, 아이콘 등
- 반응형 레이아웃: 1280x720px 최적화
- 4가지 템플릿: 표지, 목차, 콘텐츠, Q&A

**PDF 데이터 반영 내용**

- 구체적인 점수: MVP 정의 89점, 아키텍처 설계 62점 등
- 도구별 성과: Gemini, Claude, Copilot, Readdy 등
- 3개 POD 검증 결과: 실무 적용 가능 수준 판단
- Best Practice: 도구 조합 최적화 방안

발표자료가 완성되었으며, AI가 소프트웨어 개발을 주도할 수 있는지에 대한 실증적 검증 결과를 체계적으로 전달할 수 있는 구조로 구성되어 있습니다. 추가 수정이나 특정 페이지 개선이 필요하시면 언제든 말씀해 주세요!

복사  Notion에 저장

AI, 개발을 주도할 수 있을까? - KT AI Le... 저장 지점-5

미리보기 코드 생각 중 1 / 12

AI, 개발을 주도할 수 있을까?

KT AI Leading PoC 여정과 결론

2025년 7월 16일

KT

미리보기 코드 생각 중 2 / 12

# Q&A

**감사합니다.**