

Lecture 3.

프로젝트 관리 도구 및 팀 구성

김영빈 교수

학습 목표

1. 프로젝트 관리의 개요 및 필요성

- 프로젝트 관리의 개념과 필요성 명확히 이해
- 프로젝트 관리 도구의 역할과 활용 목적 이해
- 대표적인 프로젝트 관리 도구의 특징 파악

2. 팀 구성 및 역할 분담의 중요성

- 프로젝트 성공에 있어 팀 구성의 중요성 이해
- 팀 내 명확한 역할 분담이 필요한 이유 학습
- 성공적인 역할 부담 사례를 통해 주요 원칙 파악

3. Notion과 Github Issues 활용의 이론적 이해

- Notion과 Github Issues가 프로젝트 관리에 미치는 영향 이해
- 각 도구의 활용 방법과 장단점 비교
- 두 도구를 함께 사용하는 것의 장점 이론적으로 이해



GitHub

프로젝트 관리의 중요성

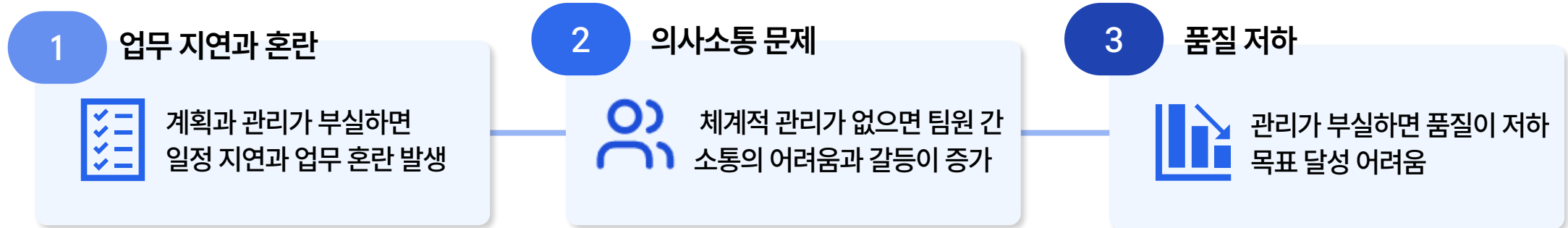
프로젝트 관리

: 목표한 결과를 얻기 위해 프로젝트의 **범위, 시간, 비용, 인력** 등을 계획하고, 체계적으로 관리 및 통제하는 활동

프로젝트 관리가 필요한 이유

- 프로젝트 목표 달성을 위한 **명확한 계획** 수립 가능
- 업무 간 우선순위를 **효과적으로 설정**하고 관리 가능
- 문제 발생 시 **빠르게 파악**하고 즉각적인 대응 가능
- 자원의 낭비를 줄이고 **생산성을 높이**는데 기여

프로젝트 관리가 제대로 이뤄지지 않을 경우 발생하는 문제



→ 성공적인 프로젝트 수행을 위해서는 효과적인 프로젝트 관리가 필수적

프로젝트 관리의 중요성

오픈소스 프로젝트와 기업 프로젝트의 관리 특성 비교

구분	오픈소스 프로젝트	기업 프로젝트
목표 및 동기	기술 공유 및 협업 중심	이윤 창출과 사업 목표 중심
팀 구성 방식	자발적, 분산적, 공개적 구성	조직 내부의 명확한 위계 구조
관리 스타일	수평적이고 자율적인 의사소통 구조	수직적이고 명확한 책임 분담 구조
소통 방식	공개된 도구 (GitHub 등) 활용	내부적, 폐쇄적 도구 활용
문제 대응 방법	커뮤니티 기반의 집단 지성 활용	체계적이고 프로세스 중심 대응

오픈소스 프로젝트 관리 특징



- 정보의 투명과 접근성이 높음
- 공개적인 피드백과 협력이 활발히 이루어짐

기업 프로젝트 관리 특징



- 명확한 역할 분담 및 보고 체계로 효율성을 높임
- 체계적인 관리 프로세스로 프로젝트 위험을 최소화

01 팀 구성과 역할 분담

팀 구성

팀(Team)

: 공통된 목적을 이루기 위해 두 명 이상이 모인 집단으로, 서로 협력하며 공동의 목표를 향해 나아가는 조직적 단위

- 개인의 능력을 초월하는 성과를 내기 위한 협력체
- 서로 보완적이고 명확한 역할과 책임을 가지며 상호의존적 관계 유지
- 목표 달성을 위해 명확한 의사소통과 협력이 필수적

◆ 오픈소스 프로젝트에서의 팀

- 명확한 목표 설정과 협력적 커뮤니케이션 필수
- 자발적 참여 기반이므로 명확한 역할 정의 중요

◆ 팀과 그룹의 차이

구분	팀(Team)	집단(Group)
목적	명확한 공동 목표 달성	개별적 목표 달성
책임	상호 의존적이며 공동 책임	독립적이며 개인별 책임
커뮤니케이션	활발한 소통 및 협력	제한적이고 선택적 소통
결과	집단적 성과 중시	개별적 성과 중시

◆ 효과적인 팀 구성의 조건

역할의 명확성

원활한 커뮤니케이션

역량의 다양성과 균형

신뢰와 존중

역할 분담 – 효과적인 역할 분담

❖ 역할 분담 시 고려사항

역할 부여 시 개인 역량 및 관심사 고려

- 팀원의 강점과 흥미, 전문 분야를 파악하여 적절한 역할 부여

커뮤니케이션 채널의 명확한 설정

- 누가 누구와 소통해야 하는지 명확하게 설정

명확한 책임과 권한의 정의

- 각 역할에 대해 명확하게 책임을 규정
- 권한과 책임 균형 잡히도록 설계


역할의 중복 방지

- 역할 충돌을 최소화하여 업무 혼선 방지


❖ 역할 분담이 필요한 이유

- 업무 중복과 혼란 방지
- 책임 소재 명확화
- 프로젝트 일정 및 품질 관리 용이
- 팀원의 전문성 극대화


❖ 역할 분담의 사례 (오픈소스 중심 사례)

 프로젝트 매니저 (PM)


- 프로젝트 전체 일정과 진행 상황 관리
- 팀원 간 커뮤니케이션 촉진 및 조정 역할 수행

 개발자 (Developer)

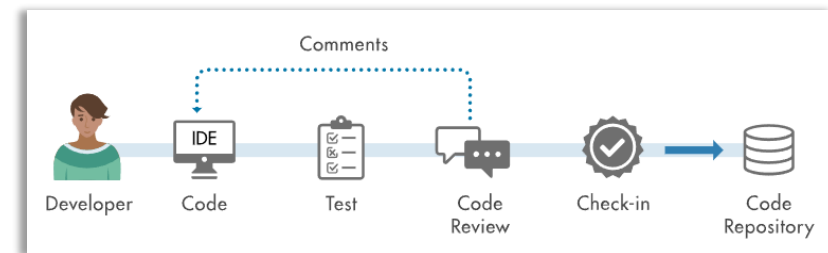
- 코드 작성 및 기술적 문제 해결 담당
- 코드 품질 유지 및 기술적 의사결정 참여

 리뷰어 (Reviewer)

- 작성된 코드의 품질과 오류 점검
- 팀 전체 코드의 일관성과 유지보수성 관리






 문서화 담당자 (Documenter)

- 프로젝트 진행 상황과 문서 작성, 관리 및 최신화
- 프로젝트의 정보 접근성 높이고 공유 용이성 유지



▲ 일반적인 개발 워크플로우

역할 분담 – 오픈소스 프로젝트 내 역할 정의

역할명	주요 역할과 담당 업무	역할이 불분명하거나 부족할 때 발생하는 문제
 PM (프로젝트 매니저)	<ul style="list-style-type: none"> 프로젝트 일정 및 진행 상황 관리 이슈 및 우선순위 설정, 커뮤니케이션 총괄 	<ul style="list-style-type: none"> 팀 전체의 방향성 혼란 발생 업무의 중복 및 우선순위 혼란
 개발자 (Developer)	<ul style="list-style-type: none"> 코드 작성 및 기능 구현 버그 수정 및 이슈 처리 제품 및 서비스 기능 개선 	<ul style="list-style-type: none"> 기능 미완성 또는 품질 저하 기술적 문제 방치로 인한 프로젝트 정체
 문서 담당자 (Documenter)	<ul style="list-style-type: none"> 프로젝트 문서 작성 및 관리 (README, Wiki 등) 프로젝트 정보 접근성 및 사용성 향상 	<ul style="list-style-type: none"> 참여자의 이해 부족으로 프로젝트 사용성 저하 신규 참여자 참여 어려움
 QA 및 리뷰어 (Quality Assurance)	<ul style="list-style-type: none"> 코드 리뷰 및 품질 점검 테스트 진행 및 버그 보고 프로젝트 품질 유지 및 개선 	<ul style="list-style-type: none"> 품질 저하 및 사용자 불만 증가 버그로 인한 프로젝트 신뢰성 하락
 커뮤니티 매니저	<ul style="list-style-type: none"> 프로젝트 커뮤니티 관리 및 활성화 사용자 피드백 수집 및 반영 	<ul style="list-style-type: none"> 커뮤니티 침체로 프로젝트 지속성 하락 피드백 부족으로 프로젝트 개선 어려움

역할 분담 – 팀 구성 사례

사례 1. 리눅스 커널(Linux Kernel) 프로젝트

리눅스 커널(Linux Kernel) 프로젝트는 전 세계 개발자들이 협업하여 운영체제의 핵심인 커널을 개발·유지 보수하는 대표적인 오픈소스 프로젝트이다.



▼ 리눅스 커널 프로젝트의 팀 구성 및 주요 역할

역할	주요 인물	역할 특징 및 업무 내용
프로젝트 리더	Linus Torvalds	<ul style="list-style-type: none">프로젝트의 전체적 방향성과 주요 기술 결정주요 변경사항 및 릴리스 최종 승인
유지 관리자 (Maintainers)	서브시스템 별 책임자들 (e.g. Greg Kroah-Hartman 등)	<ul style="list-style-type: none">커널 내 각 서브시스템 책임 관리 및 유지코드 병합(Merge) 승인 및 품질 관리
기여자 (Contributors)	전 세계 수천 명의 개발자	<ul style="list-style-type: none">코드 작성, 버그 수정, 기능 구현리뷰 요청 및 피드백 반영
리뷰어 (Reviewers)	숙련된 개발자 및 유지 관리자 그룹	<ul style="list-style-type: none">코드 품질 검증 및 기술적 피드백 제공유지 보수성과 안정성 확보

✓ 성공 요인

- 명확한 역할과 권한 분배
- 신뢰와 책임 기반의 자율적 관리
- 투명한 소통 구조로 활발한 협력 촉진

역할 분담 – 팀 구성 사례

사례 2. 텐서플로우(TensorFlow) 프로젝트

텐서플로우(TensorFlow) 프로젝트는 구글이 개발한 오픈소스 머신러닝 라이브러리로, 체계적인 역할 분담을 통해 빠르게 발전한 성공적인 프로젝트이다.

▼ 텐서플로우 프로젝트의 팀 구성 및 주요 역할

역할	담당자 및 그룹	역할 특징 및 업무 내용
제품 총괄 (PM 및 Technical Lead)	구글 내부 프로젝트 매니저 및 리드 개발자	<ul style="list-style-type: none">프로젝트 전체 로드맵과 기술 방향 결정주요 기능 우선순위 설정 및 관리
핵심 개발자 (Core Developers)	구글 내부 및 오픈소스 핵심 개발자 그룹	<ul style="list-style-type: none">라이브러리 핵심 코드 구현 및 유지 관리주요 기술적 의사결정 참여
기술문서 팀 (Documentation Team)	내부 전담 팀 및 커뮤니티 참여자	<ul style="list-style-type: none">사용자 가이드 및 API 문서 작성 및 관리신규 참여자의 프로젝트 접근성 향상
QA 및 테스트 팀 (Testing Team)	내부 개발자 및 오픈소스 커뮤니티 테스터	<ul style="list-style-type: none">자동화된 테스트 관리 및 버그 추적코드 안정성 유지 및 배포 품질 보장
커뮤니티 지원팀 (Community Support)	구글 직원 및 오픈소스 활동가	<ul style="list-style-type: none">GitHub Issues 및 StackOverflow를 통한 사용자 지원프로젝트 피드백 및 의견 수렴



TensorFlow

✓ 성공 요인

- 체계적인 관리와 명확한 책임 분담
- 기술 문서화를 통한 사용성 향상 및 접근성 강화
- 적극적인 커뮤니티 지원으로 지속적 사용자 유입 및 유지

역할 분담 – 역할 분담 시 발생하는 문제

문제 1. 중복된 업무 발생

정의 및 원인

- 역할 정의가 모호업무 공유 및 커뮤니케이션 부족으로 서로의 작업 내용을 인지하지 못할 때 발생

실제 사례

- GitHub Issues가 명확히 할당되지 않아 여러 명이 같은 버그를 동시에 수정하고 충돌 발생

부정적 영향

- 자원의 비효율적 사용 및 시간 낭비
- 팀원 간 업무 갈등 및 프로젝트 진행 혼란 초래

문제 2. 책임 회피 현상

정의 및 원인

- 책임과 역할이 모호할 때, 팀원이 어려운 작업이나 문제 해결을 서로 미루는 현상

실제 사례

- 긴급한 버그가 발생했으나 담당자가 명확하지 않아 아무도 대응하지 않다가 문제가 확대된 경우

부정적 영향

- 업무 진행 지연 및 품질 저하
- 장기적으로 프로젝트 실패 위험 증가

문제 3. 업무 겹침 및 역할 충돌

정의 및 원인

- 업무 영역이나 역할 범위가 겹쳐 서로의 업무를 침범하거나 충돌하는 현상

실제 사례

- PM과 개발자가 서로의 의사결정 권한을 침범해 기술적 결정이 지연되거나 충돌한 경우

부정적 영향

- 업무 혼선, 역할 충돌로 인해 팀 내 갈등 유발
- 프로젝트의 전반적인 효율성 저하 및 커뮤니케이션 악화

역할 분담 – 역할 분담 문제의 해결 방법

❖ 역할 분담 문제의 해결 방법

해결 방법 1. 명확한 역할 정의 및 문서화

- 역할 정의 문서를 모든 팀원이 접근 가능한 도구(Notion, GitHub Wiki)에 정확히 게시하여 상시 열람 가능하도록 관리

해결 방법 2. 투명하고 지속적인 커뮤니케이션 활성화

- 정기적인 회의를 통한 공유
- GitHub Issues, Notion 등을 활용하여 실시간으로 업무 상태 업데이트 및 업무 분배 구체화

해결 방법 3. 업무 추적 시스템 도입 (Issue Tracking)

- 프로젝트 관리 도구(GitHub Issues 등)를 통해 각 업무의 할당자, 우선순위, 기한을 명확히 설정하고 지속적으로 관리

해결 방법 4. 주기적인 역할 재검토와 피드백 제공

- 프로젝트 진행 중 역할 분담이 적절한지 정기적으로 검토하고 조정 (역할 조정 회의)

❖ 성공적인 프로젝트 수행을 위한 효과적인 팀 구성과 관리방법

1. 명확하고 구체적인 역할 정의

- 팀원 개인의 역할과 책임을 명확히 정의
- 역할 정의를 반드시 문서화

2. 지속적이고 명확한 커뮤니케이션 환경 조성

- 투명한 소통 채널 마련 (정기 회의, 협업 도구 활용 등)
- 모든 업무 현황을 실시간으로 공유

3. 업무 진행 상황 지속적 관리 및 피드백

- Issue 관리 도구를 적극적으로 활용하여 업무 상태와 진척도를 지속적으로 관리
- 주기적으로 진행 상황을 점검 및 신속한 피드백을 제공

4. 팀 구성 시 다양한 역량과 균형 확보

- 팀 구성 시 각 팀원의 강점과 약점을 파악하여 균형 잡힌 역할 분배
- 전문성과 관심사에 맞춘 역할 분담으로 업무 만족도와 생산성을 높임
- 역량의 다양성이 프로젝트의 창의성과 안정성을 높이는 데 기여

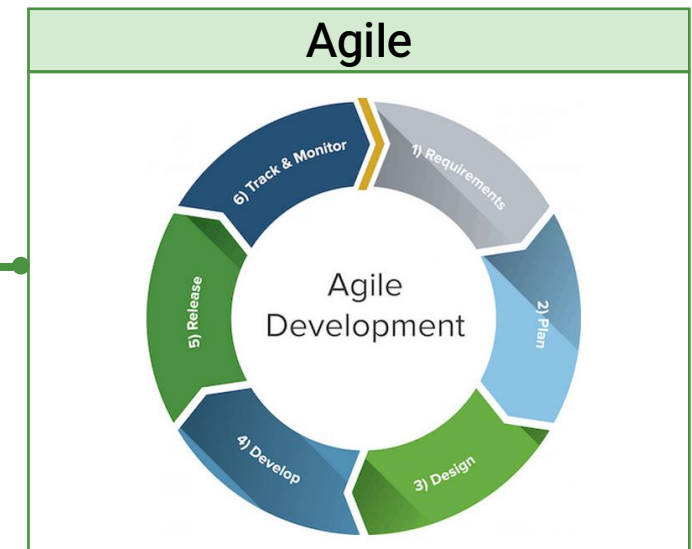
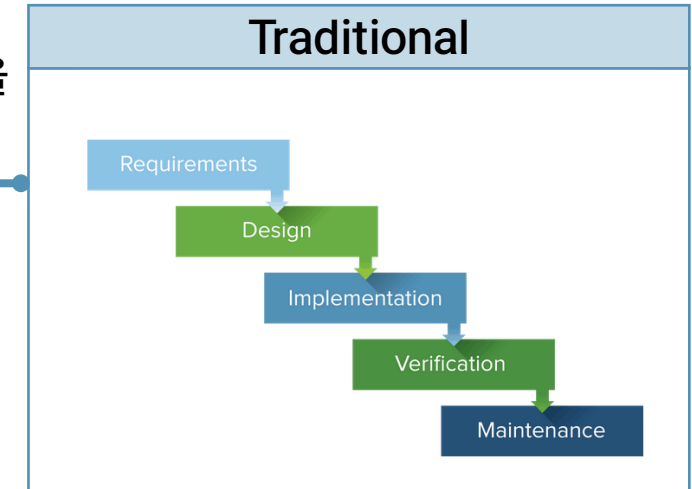
02 프로젝트 관리 도구의 이론적 개요

프로젝트 관리 도구

프로젝트 관리 도구(Project Management Tools)

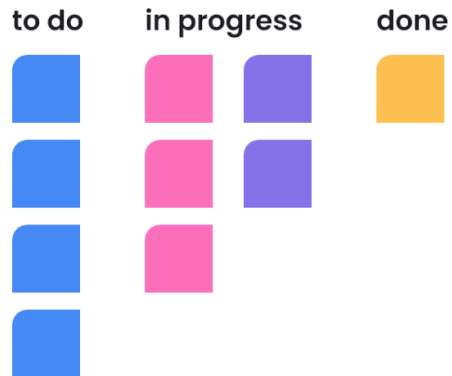
: 프로젝트의 목표를 효율적이고 성공적으로 달성하기 위해 업무의 계획, 일정, 자원 배분, 진행 상황 등을 체계적으로 관리하고 통제할 수 있도록 지원하는 소프트웨어 및 시스템

구분	전통적(Traditional) 관리 도구	애자일(Agile) 관리 도구
목적 및 특성	명확한 일정과 업무 범위에 따라 사전 계획 및 관리 중심	유연한 변화와 빠른 대응 중심의 점진적 관리 방식
관리 방식	선형적(Linear), 순차적(Sequential)	반복적(Iterative), 점진적(Incremental)
대표 도구	Microsoft Project, Excel, Gantt 차트	Kanban, Scrum, Jira, Trello, Notion 등
업무 환경	계획이 명확하고 변화가 적은 프로젝트	변화가 빈번하고 불확실성이 높은 프로젝트
의사소통 방식	문서 중심의 공식적 의사소통	실시간 소통 및 지속적 피드백 중심
특징	<ul style="list-style-type: none"> 계획 중심의 접근 프로젝트 시작 시점에 상세한 계획 수립 및 확정 명확한 범위와 일정 정해진 목표와 일정을 엄격히 지키는 관리 방식 엄격한 통제와 감독 일정, 비용, 범위를 철저히 관리하고 변화 최소화 	<ul style="list-style-type: none"> 유연성과 신속한 대응 중시 프로젝트 진행 중 발생하는 변화를 자연스럽게 반영하고 신속히 대응 지속적인 피드백과 반복 작업 주기적 반복(iteration)을 통해 지속적인 개선 및 개발 진행 자율적이고 분산된 업무 진행 팀원 간 자율적 소통과 협력을 기반으로 업무 수행



프로젝트 관리 도구 : Agile

1. 칸반(Kanban) : 업무 흐름을 시각적으로 나타내고 업무 상태를 명확히 관리하는 방식



특징

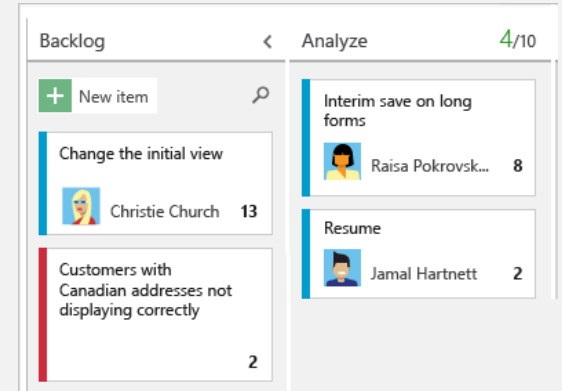
- 업무 진행 상황을 카드 형태로 표현하여 시각화
- 작업 중(Work in Progress, WIP) 개수를 제한하여 업무 과부하 방지

장점

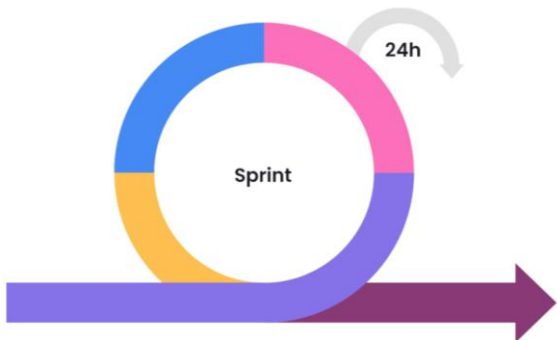
- 업무 투명성 확보 및 문제점 신속히 발견 가능
- 자율적 관리 및 실시간 대응 가능

주요 도구 예시

- Notion Kanban Board, Trello, Jira Kanban 등



2. 스크럼(Scrum) : 일정 기간(스프린트, Sprint)을 정하고, 반복적으로 목표 달성을 위해 업무를 진행하는 방식



특징

- 명확한 기간 내에 명확한 목표 달성을 위한 반복(Iteration) 중심
- 빠른 반복을 통해 지속적으로 프로젝트를 개선하는 Agile 방법론의 대표적인 방식
- 역할(PM, 개발자, Scrum Master)이 명확하게 설정되고 협력적으로 운영

장점

- 빠른 피드백을 통한 신속한 개선 가능
- 변화에 유연하게 대응하며 목표에 빠르게 접근 가능

주요 도구 예시

- Jira Scrum, Azure DevOps, GitHub Project 등

스크럼 주요 활동 예시

스프린트 계획 미팅 → 업무 할당 및 목표 설정
데일리 스크럼 미팅 → 업무 진행 상황 매일 공유

프로젝트 관리 도구 – 도구 선택 기준

❖ 오픈소스 프로젝트 특성상 도구 선택 시 반드시 고려할 사항

접근성과 투명성 (Accessibility & Transparency)

- 모든 참여자가 프로젝트 정보를 쉽게 접근할 수 있어야 함

협업의 용이성 (Collaboration Ease)

- 전 세계의 참여자가 쉽게 업무를 공유하고 소통할 수 있어야 함



정보 통합성 및 관리 효율성 (Integration & Efficiency)

- 문서, 코드, 이슈, 일정 등의 정보를 하나로 통합하여 관리 가능한지 고려

유연성과 확장성 (Flexibility & Scalability)

- 프로젝트의 변화와 성장에 따라 도구의 기능이 확장 가능해야 함

❖ 오픈소스 프로젝트에서의 GitHub와 Notion

도구	주요 활용 분야	장점
 GitHub	코드 관리, 협업 및 이슈 관리	<ul style="list-style-type: none">• 코드 중심 협업 특화• 투명하고 공개적인 커뮤니케이션 환경 제공
 Notion	프로젝트 문서화, 정보 통합 관리	<ul style="list-style-type: none">• 다양한 유형의 정보를 효율적으로 통합 관리• 유연하고 확장 가능한 데이터베이스 관리 지원

프로젝트 관리 도구 – GitHub와 Notion의 적합성 비교



GitHub가 적합한 경우

주요 장점

- 코드 중심 프로젝트에 최적화된 강력한 Git 기반 관리
- 이슈 트래킹, PR, Discussions 등 협업 기능 제공
- 코드 리뷰 및 피드백이 활발한 환경에 적합
- 오픈소스 커뮤니티와 연계 용이, 신규 참여자 유입 원활

한계

- 문서화 및 데이터 관리에 제한적
- 비정형 정보 관리 어려움, 프로젝트 정보 분산 가능

Notion이 적합한 경우



주요 장점

- 문서화, 자료 공유, 일정 및 업무 관리에 최적화
- 페이지·블록 기반의 유연한 정보 구성
- 강력한 데이터베이스 기능으로 정보 통합 관리 용이
- 직관적인 UI로 비기술적 참여자도 쉽게 활용 가능

한계

- 코드 호스팅 및 버전 관리 지원 부족
- PR 기반 협업 및 실시간 공개 리뷰에 제한적

목적	추천 도구	실제 활용 사례
코드 관리 및 기술적 협력 →	GitHub	<ul style="list-style-type: none"> • 리눅스 커널 개발 프로젝트 • TensorFlow 개발 및 협업 프로젝트
문서화 및 관리 정보 통합 →	Notion	<ul style="list-style-type: none"> • 다양한 스타트업 및 기업의 프로젝트 관리 • 오픈소스 커뮤니티의 로드맵 및 정보 공유 페이지
두 가지 목적 모두 필요 →	GitHub + Notion 연계	<ul style="list-style-type: none"> • GitHub Issues와 Notion을 연계하여 프로젝트 정보 효율적으로 관리 • 코드 관리와 프로젝트 문서화를 동시에 수행하는 최근 오픈소스 프로젝트 경향

◀ GitHub와 Notion
선택 가이드

03 Notion의 이론적 이해

Notion의 개요



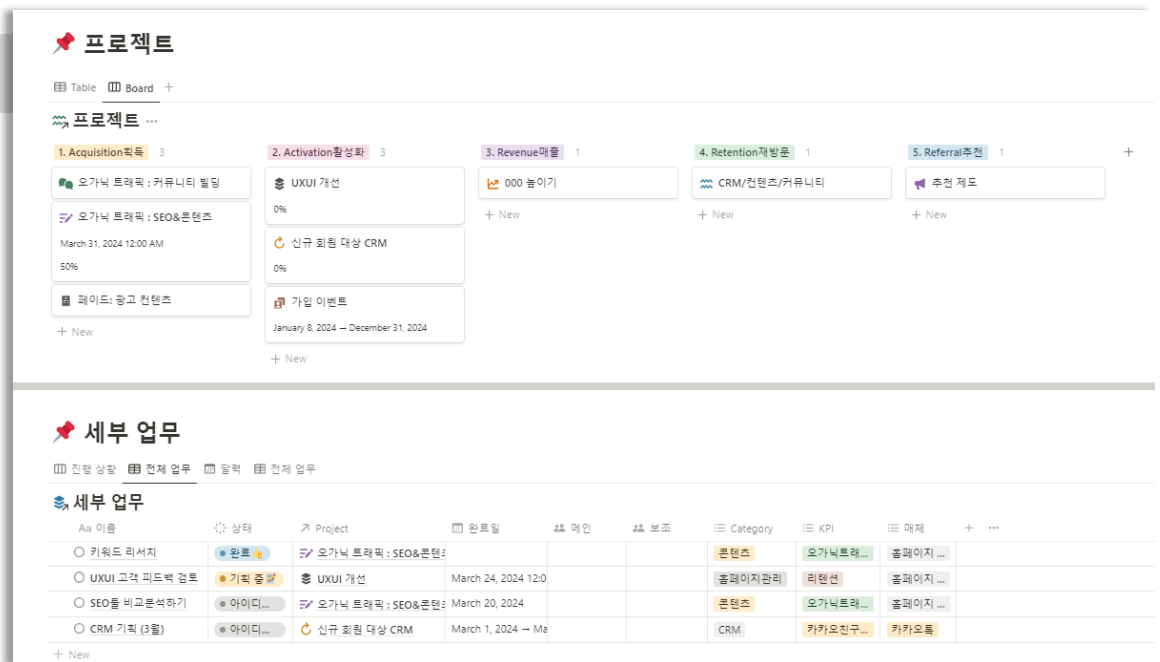
Notion

- 노트 작성, 문서화, 업무 관리, 데이터베이스 구축 등 **다양한 작업을 하나의 틀에서 통합적으로 수행할 수 있게 만든 생산성 도구**
- 기존에 여러 앱으로 나뉘어 있던 기능(워드, 스프레드시트, 캘린더, 업무관리 등)을 하나로 통합하여 제공하며, 사용자가 유연하게 정보를 관리할 수 있음

▼ 협업에서 Notion을 사용하는 예시

❖ 현재 Notion의 위치와 역할

- **개인 생산성 도구**
개인의 지식 관리, 일정 관리, 업무 관리 등 다양한 생산성 향상을 위한 도구로 활용
- **기업 협업 도구**
스타트업 및 기업에서 협업, 문서화, 프로젝트 관리를 위한 필수적인 도구
- **오픈소스 프로젝트 관리 도구**
프로젝트 관리 및 정보 공유를 위한 표준 도구로 활용



Notion의 핵심 구조 : Block

1. 블록(Block)

: 정보를 담는 Notion의 기본적인 단위로, 레고 블록처럼 자유롭게 쌓고 조합하여 정보를 구성.
모든 콘텐츠는 개별 블록으로 구성 (유연한 콘텐츠 관리 가능)

❖ 블록 기반 구조의 장점

- 콘텐츠의 배치 및 재구성이 매우 유연하고 직관적
- 정보를 시각적으로 구성하고 관리하기 용이하여 사용성과 접근성이 뛰어남

블록의 대표적인 예시

텍스트 블록
기본 텍스트, 제목, 목록, 인용 등 다양한 텍스트 형태

할 일 블록(To-do)
체크박스 형태의 할 일 관리

코드 블록
프로그래밍 언어로 작성된 코드 표시 및 관리

데이터베이스 블록
표(Table), 칸반(Kanban), 캘린더(Calendar) 등 구조화된 정보 관리

미디어 블록
이미지, 동영상, PDF 등 멀티미디어 자료

Notion의 핵심 구조 : Page

2. 페이지(Page)

: 여러 개의 블록을 담을 수 있는 공간으로, 하나의 문서 또는 하나의 주제를 담는 독립된 작업 공간

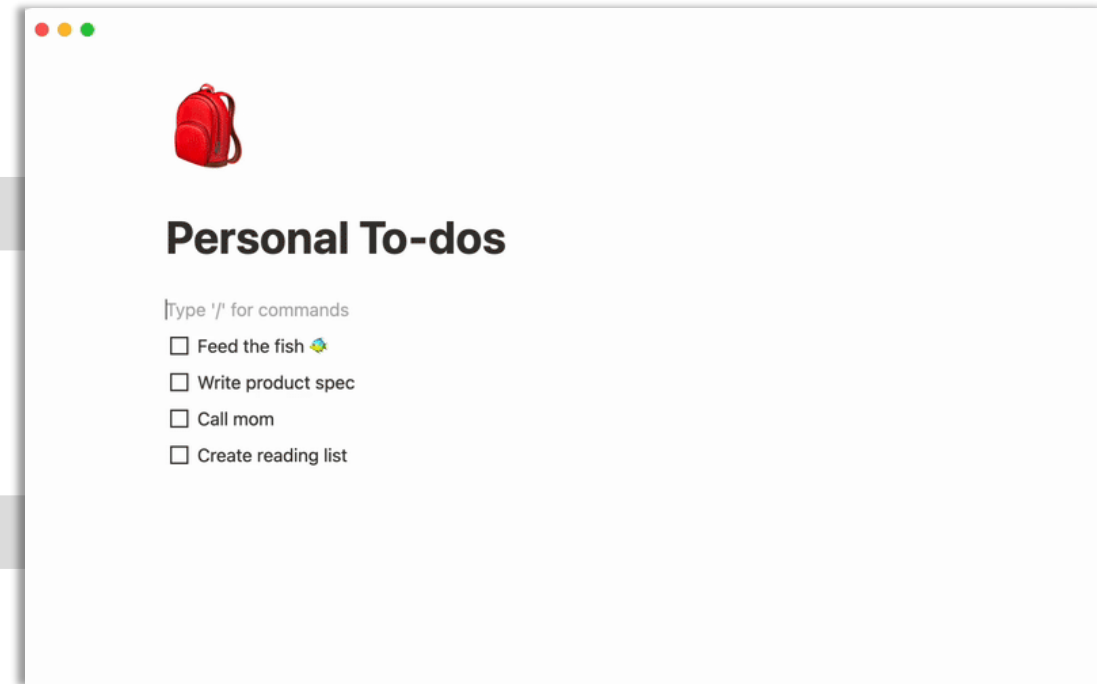
- 페이지는 자유롭게 하위 페이지(Sub-page)를 생성
- 계층적 구조화 및 정보 확장에 용이
- Notion에서 모든 블록은 페이지라는 더 큰 단위 안에서 관리

❖ 페이지의 개념적 특징

- 계층적 구조화 가능
- 링크와 참조의 용이성
- 다양한 형식의 콘텐츠를 혼합 가능

❖ 블록 기반 구조의 장점

- 유연한 정보 관리
- 뛰어난 확장성
- 정보 접근성 향상



▲ 페이지 내에서 원하는 블록을 추가하는 과정

Notion의 핵심 구조 : Database

3. 데이터베이스(Database)

: Table 형태로 구조화된 데이터를 저장하고, 다양한 방식으로 관리할 수 있는 기능

- 일반 페이지와 달리 구조화된 정보를 체계적으로 관리할 수 있음
- 데이터를 쉽게 검색, 필터링, 정렬 가능
- 동일한 정보를 다양한 뷰로 전환하여 볼 수 있음

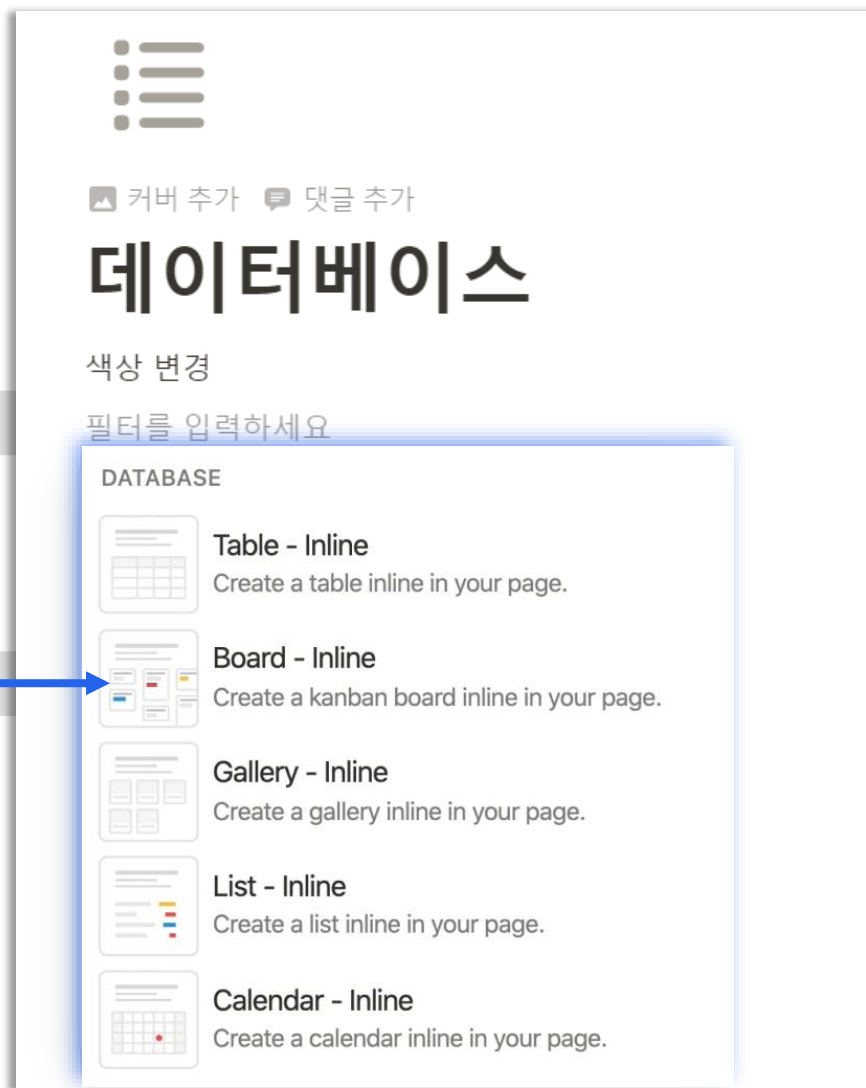
→ 스프레드시트와의 유사점

Notion의 데이터베이스는 스프레드시트처럼 표 형태로 정보를 관리하되,
더 다양한 형태로 시각화 가능

→ 다양한 뷰 전환 기능

테이블, 캘린더, Kanban, 리스트, 갤러리 등 다양한 뷰로 바꿔볼 수 있는 유연성 제공

Notion에서의 Database ►



Notion의 핵심 구조 : Database

◆ Notion Database의 활용

테이블(Table) 뷰

- 데이터를 표 형태로 관리하며 필터링 및 정렬 기능 제공
- 활용: 프로젝트 업무 리스트, 버그 추적

CRM

Table View ▾

Properties Filter Sort Search ... New ▾

Name	Company	Status	Priority	Account Owner	Estimated Value	@ Em
Kim Saunders	Summly	Proposal	Medium	Cory Etzkorn	\$30,000.00	ks12@
Mary Meeks	Bark	Lead	Low	Leslie Jensen	\$20,000.00	maryr
Mike Mendez	Frontier Tech	Negotiation	Low	Shirley Miao	\$30,000.00	mike@
Edwin Chan	Tims	Closed	Medium	Harrison Medoff	\$50,000.00	edwin

캘린더(Calendar) 뷰

- 날짜 기반 데이터 관리 및 일정 시각화 제공
- 활용: 프로젝트 일정, 마일스톤 관리



Calendars

Calendar 1

February 2020						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
						Feb 1
2	3	4	5	6	7	8

리스트(List) 뷰

- 정보를 목록 형태로 나열하여 간단한 데이터 상태 관리 가능
- 활용: 할 일 목록, 아이디어 정리

Docs

- All Recently edited My docs PRDs +
- Recruiting new support reps
 - TikTok boosting campaign retro
 - Acme's product design system
 - TextBlock API endpoint
 - Error codes at Acme
 - Q2 Lifecycle marketing report
 - PRD: Improving desktop app load times with lazy loading
 - RFC: Marketing Asset Library

칸반(Kanban) 뷰

- 작업의 상태를 시각적으로 표현하여 업무 흐름 관리
- 활용: 업무 상태 관리, 업무 모니터링

Meeting Notes

Board All Architecture Overviews 6 more...

Filter Sort Search ... New ▾

CX 3	Engineering 3	Product 3
<ul style="list-style-type: none">Tools for accessibilityEdgar DegasPierre-Auguste RenoirMary Cassatt	<ul style="list-style-type: none">Eng Weekly 02-22-2021Edgar DegasMarie BracquemondBug tracking process	<ul style="list-style-type: none">Chat functionalityClaude MonetMarie BracquemondUX interviews

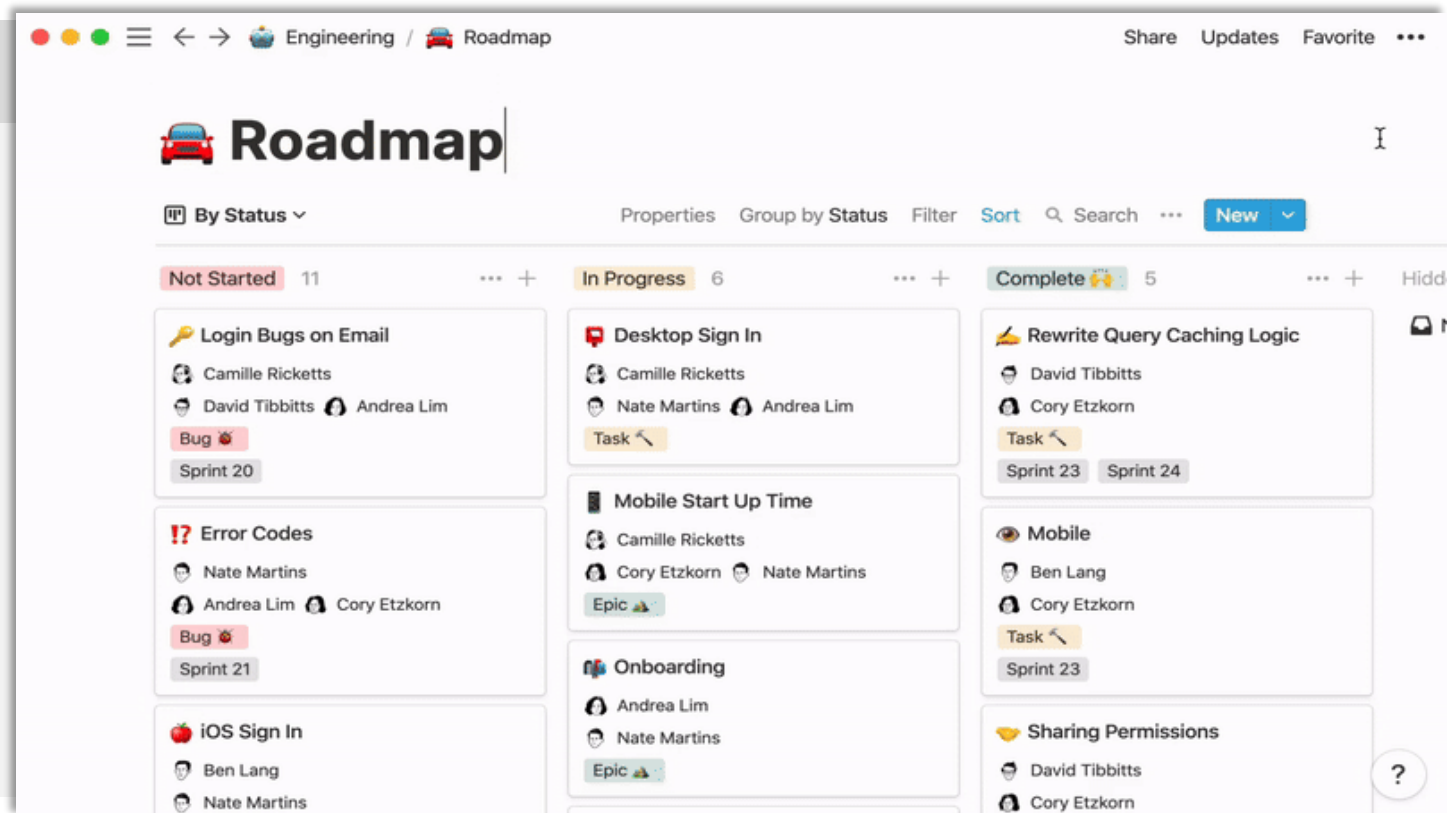
Notion의 핵심 구조 : Kanban Board

4. Kanban Board(칸반 보드)

- 작업의 시각화 및 효율적인 흐름 관리를 위한 프로젝트 관리 도구
- 드래그 앤 드롭 인터페이스로 작업 카드를 열 간에 이동시키며 업무 흐름을 직관적으로 관리

◆ 프로젝트 관리에서의 Kanban Board 활용 효과

- 유연하고 직관적인 인터페이스
: 프로젝트의 현재 상태를 직관적으로 확인 가능
- 통합된 작업 공간
- 드래그 앤 드롭 인터페이스
: 작업 카드를 열 간에 손쉽게 이동시켜 업무의 진행 상황을 업데이트 할 수 있음
- 업무 흐름 시각화 : 작업의 상태를 '할 일', '진행 중', '완료' 등의 열로 구분하여 팀과 이해관계자들이 현재 진행 상황을 쉽게 파악하고 공유
- 실시간 추적 : 팀원들이 각자의 작업 상태를 실시간으로 추적하며 조정 가능

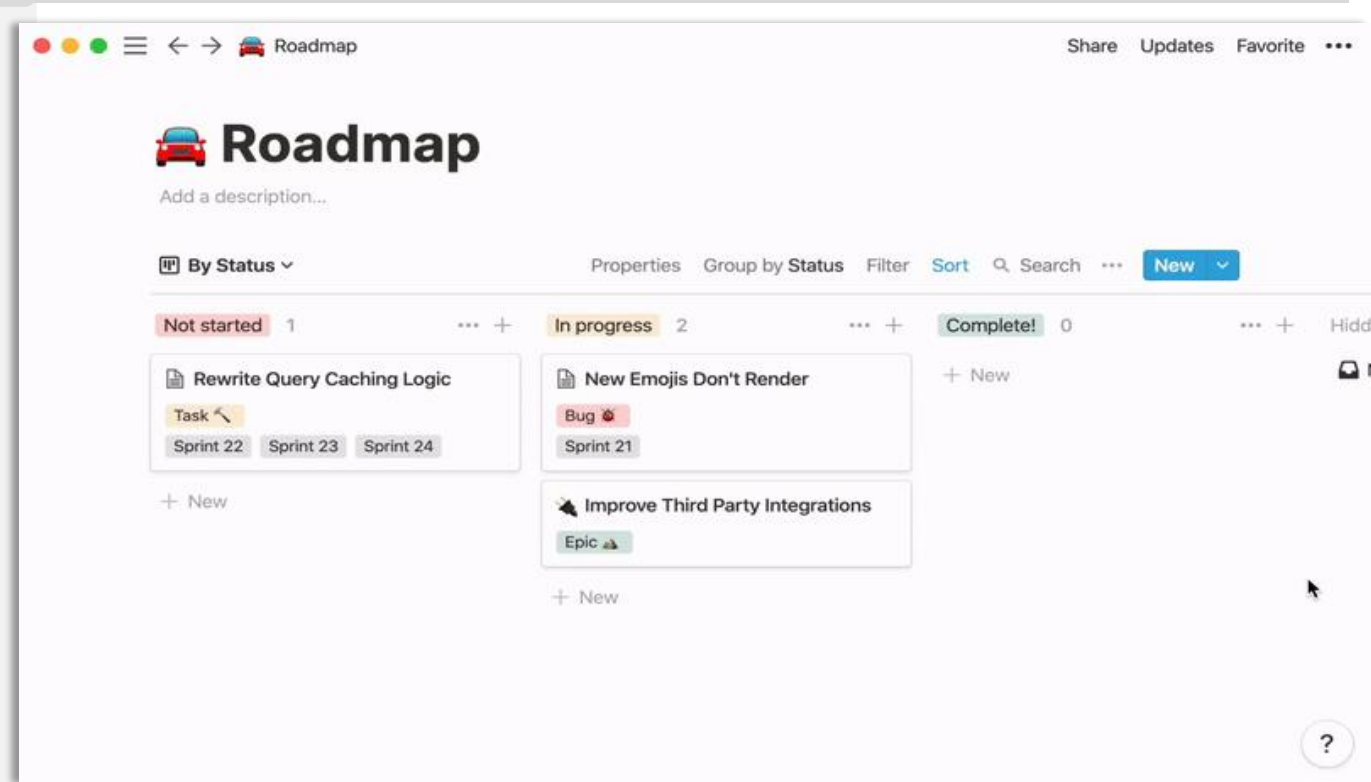


▲ Notion에서 Kanban Board로 Workflow를 작성하는 과정 예시

Notion의 핵심 구조 : Kanban Board

◆ Notion에서 Kanban Board를 만드는 방법

1. Notion의 팀스페이스 열기
2. 새 데이터베이스 생성
: 페이지에서 /데이터베이스 인라인을 입력하여 새로운 데이터베이스를 만듦
3. 작업 또는 마일스톤 추가
: 각 작업이나 마일스톤에 대한 항목을 데이터베이스에 입력
4. 작업 속성 커스터마이징
: 상태, 담당자, 스프린트 등을 포함한 작업 속성을 설정하고, 작업을 '할 일', '진행 중', '완료', '백로그', '보류 중' 등으로 지정
5. 보드 뷰 생성
: 데이터베이스를 시각화하기 위해 보드 뷰를 생성하고, 칸반 카드를 다양한 열로 이동시켜 진행 상황을 추적



▲ Notion에서 Kanban Board를 만드는 과정 예시

기업에서의 Notion 활용 사례

사례 1. Notion을 활용한 칸다(QANDA)

QANDA (칸다)

- 모르는 문제를 촬영하면 5초 안에 풀이와 맞춤형 학습 콘텐츠 제공하는 AI 기반 학습 플랫폼
- Notion을 활용한 프로젝트 관리 시스템으로 업무 복잡도를 낮추고 생산성 향상

1 Kanban Board 사용

- Notion의 Kanban Board를 활용해 1~2주 단위의 문제(에픽) 해결 과정 시각화

2 데이터를 직관적으로 시각화하는 데이터베이스

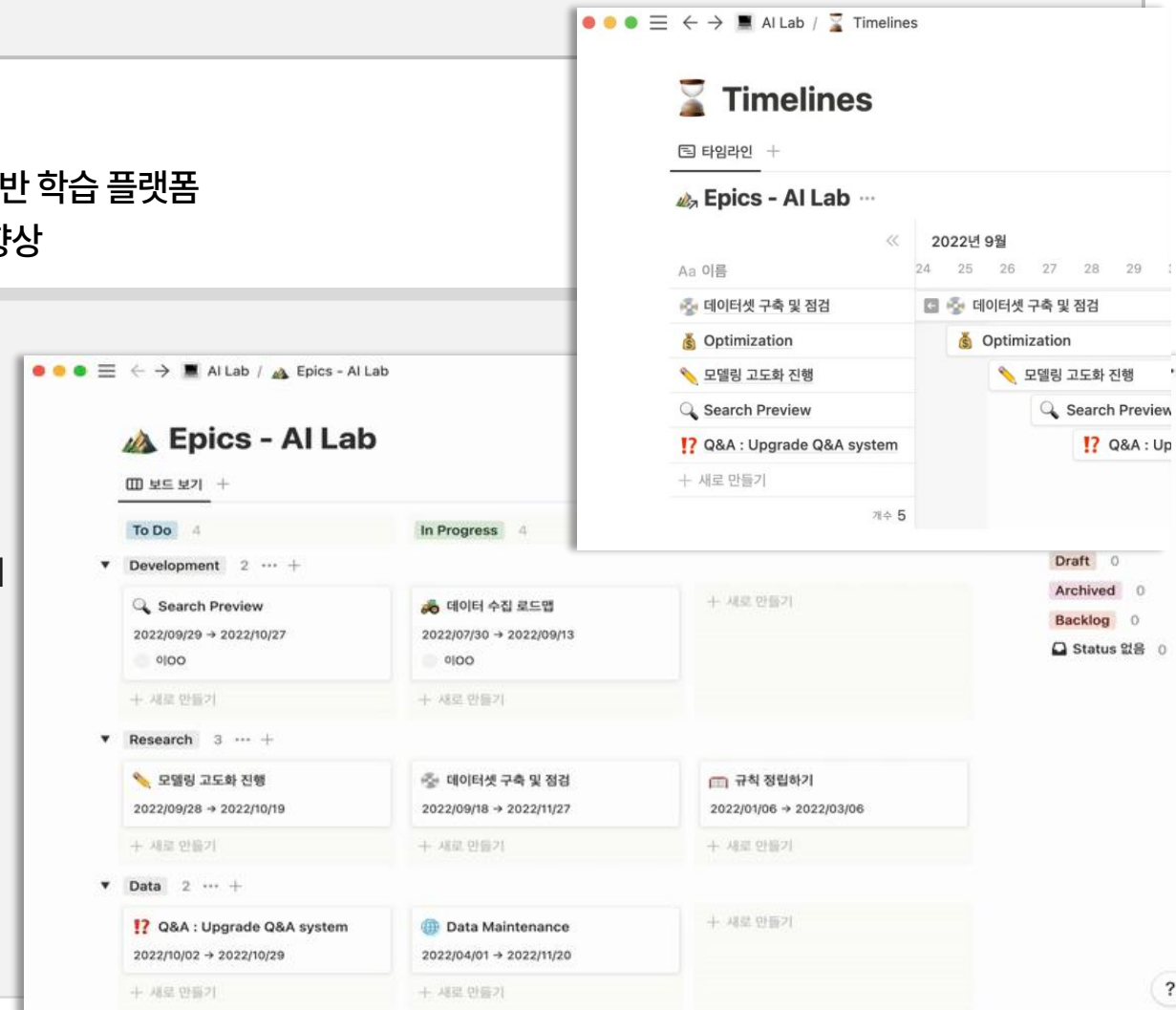
- 데이터베이스의 다양한 뷰(Kanban Board, 타임라인)를 활용해 연구·개발 데이터 정리

3 동기화로 완성하는 정교한 로드맵

- 하나의 로드맵 페이지에 모든 업무 정보를 통합하여 최신 데이터 유지

4 커스터마이징으로 효율적인 업무 루틴 구축

- 개인별·팀별 업무 방식에 맞춰 Notion을 자유롭게 커스터마이징



기업에서의 Notion 활용 사례

사례 2. 스타트업 제품 개발 및 로드맵 관리



- 아파트, 발전소, 화학 플랜트 등 다양한 건설 프로젝트를 수행하는 대한민국 대표 건설기업
- Notion을 도입하여 현장 안전 관리부터 임원 보고까지 업무 효율성과 협업 수준 향상

1 프로젝트 대시보드를 통한 현장 안전 및 품질 관리

- 건설 현장의 핵심 업무를 디지털화하여 실시간 협업 시스템 구축

2 현장 맞춤 템플릿 제작으로 효율적인 업무 프로세스 구축

- 각 건설 프로젝트 특성에 맞춘 핵심 템플릿(2~3개) 제공으로 도입 장벽 최소화

3 자발적인 프로젝트 관리를 유도하는 교육 시스템

- AI 기능(문서 요약, 데이터 자동화) 활용

4 효율적인 임원 보고 시스템 구축 및 커뮤니케이션 혁신

- 대면 보고 대신 Notion을 활용하여 실시간 피드백과 문서 공유 가능

The image displays two Notion workspace pages. The left page, titled '고위험 작업' (High-Risk Work), features a table with columns for 작업일 (Work Date), 업체명 (Company Name), 공구 (Tool), 위치 (Location), and PTW대상 (PTW Target). The table lists several construction projects with dates ranging from 2025/01/31 to 2025/07/31. The right page, titled '가나다 현장 템플릿 페이지' (Ganada Site Template Page), shows a dashboard with sections for '우선 적용 양식' (First Application Template), '주요자료' (Main Materials), and a list of templates for different site types.

작업일	업체명	공구	위치	PTW대상
2025/07/31	가나다건설	1공구	우수저류조	
2025/05/30	ABC건설	1공구	112동	
2025/03/28	ABC건설	1공구	101동	
			117동	
2025/03/01	가나다건설	1공구	111동	✓
2025/02/28	ABC건설	1공구	116동	✓
2025/01/31	가나다건설	1공구	115동	
2025/01/31	가나다건설	1공구	116동	

04 GitHub Issues

GitHub Issues

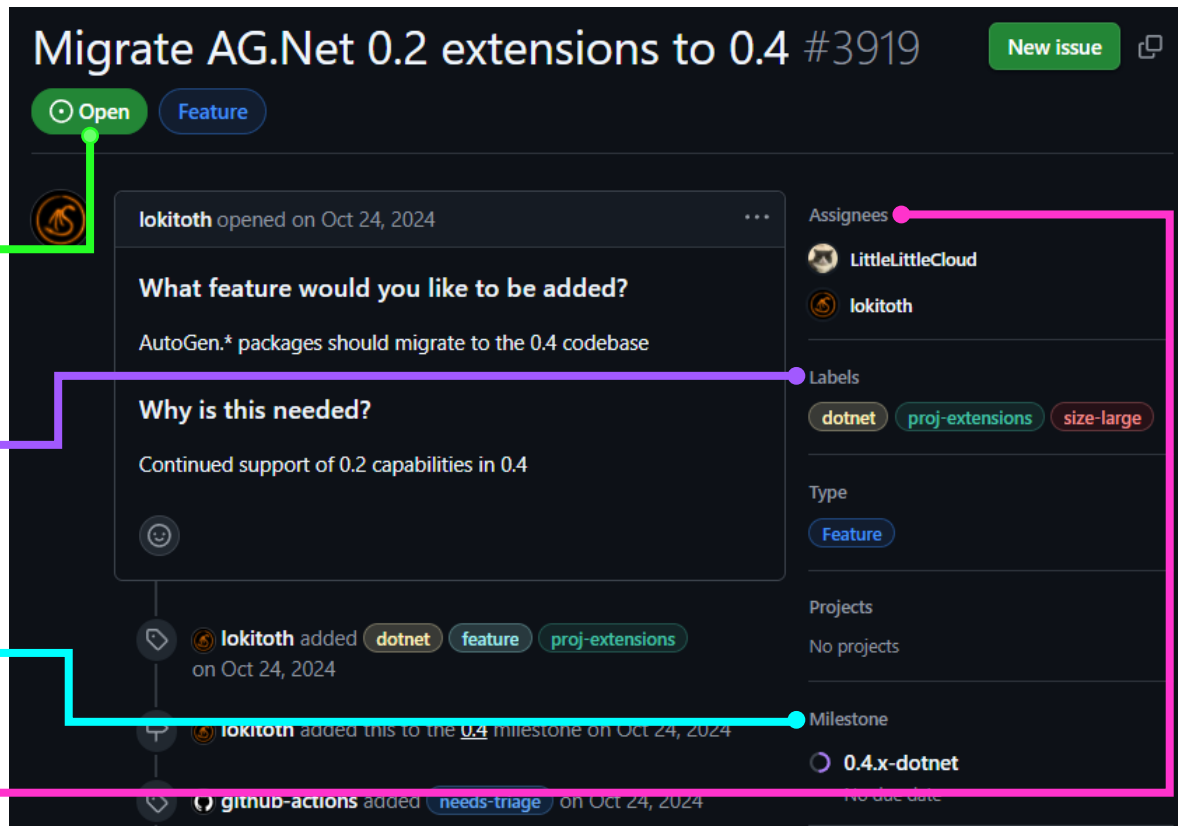
GitHub Issues

: 프로젝트의 작업(Task), 버그(Bug), 기능 요청(Feature Request), 질문(Q&A) 등을 관리하는 **커뮤니케이션 및 협업 도구**

- 프로젝트의 작업 항목이나 문제점을 명확히 정의하고 기록하는 공간
- 팀원과 사용자 간의 공개적 커뮤니케이션을 촉진
- 프로젝트의 상태를 투명하게 공유할 수 있는 도구
- 각 이슈는 독립적으로 관리, 상태(열림/닫힘) 및 우선순위 설정, 담당자 지정

▼ GitHub Issues 사용 예시

❖ GitHub Issues의 기본 구조



GitHub Issue-driven

Issue-driven Management : GitHub에서 Issue가 처리되는 흐름

: 모든 작업을 Issue 단위로 나누어, 해당 Issue 중심으로 문제를 해결하며 프로젝트를 운영하는 방식

❖ Issue-driven 관리가 주목받는 이유

작업 명확성 증가

- 업무 단위가 작아 쉽게 관리 가능
- 누가, 언제, 무엇을 해야 하는지 정의

커뮤니케이션과 협력 활성화

- 특정 Issue에 대해 팀원들이 의견을 주고받을 수 있는 구조 제공
- 투명한 커뮤니케이션을 통해 효율적으로 협업 가능

이슈 추적과 기록의 용이성

- 문제나 요청 가능이 기록으로 남아 프로젝트 히스토리를 관리하고 후속 대응이 쉬움
- 업무의 누락과 중복을 방지

업무 우선순위 설정과 관리 효율성

- 중요도 및 긴급도를 기준으로 우선순위를 정하고 관리 가능
- 라벨, 마일스톤 등을 활용하여 업무 그룹화와 체계적 관리 가능

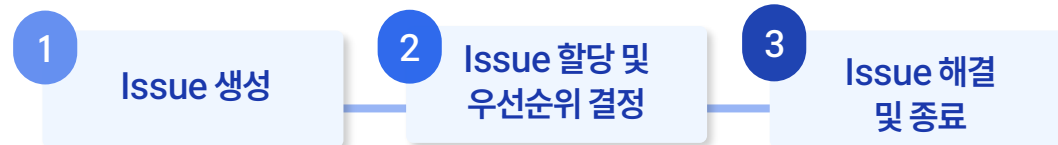
❖ Issue-driven 관리 방식의 장점

1. 명확하고 세부적인 업무 관리 기능
2. 실시간 협업과 투명한 커뮤니케이션 촉진
3. 업무 진행 상황 추적 용이성 향상

❖ Issue가 팀 협업에 미치는 긍정적 효과

- 명확한 업무 부담을 통한 협업 효율성 증대
- 투명한 정보 공유 및 커뮤니케이션 활성화
- 문제 해결 및 의사 결정 과정의 기록화

❖ Issue-driven 프로젝트 관리 프로세스

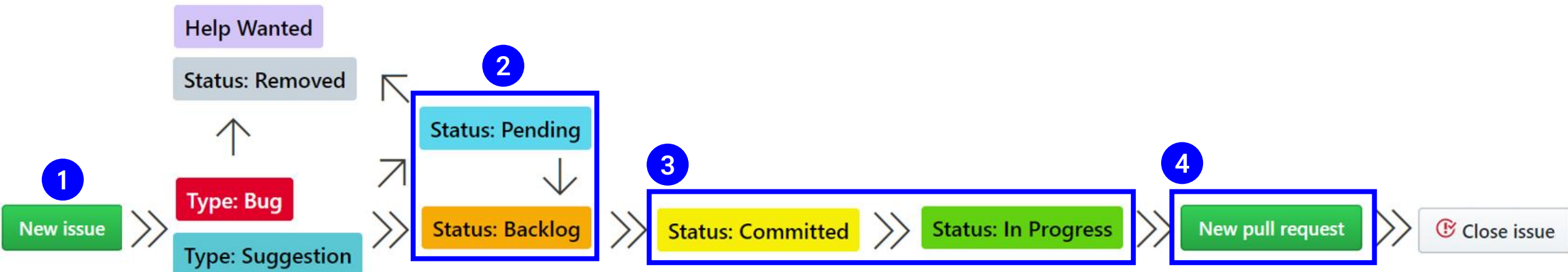


GitHub Issue-driven

◆ Issue Lifecycle

- 1 Issue Creation : 프로젝트 내에서 해결해야 하는 문제, 작업, 기능, 요청 등을 명확히 작성하여 등록
- 2 Issue 할당 및 우선 순위 지정 : Issue를 해결할 담당자(Assignee)를 지정하고 업무의 우선순위 설정
- 3 Issue 진행 및 소통 : 담당자는 할당된 Issue를 수행하고, 진행 상태를 지속적으로 업데이트
- 4 Issue 해결 및 종료 : 작업이 완료되면 결과 등록 후 Issue를 Close

▼ Issue-driven Management Workflow





GitHub Issue-driven

▼ Issue 관리 예시


Issue Lifecycle	방법	예시
❶ 생성	사용자가 “로그인 오류” 보고	제목 : “로그인 시 오류 발생”
❷ 할당	PM이 해당 문제를 우선순위 높음으로 설정	담당자 : 김민수 (백엔드 개발자)
❸ 진행	담당 개발자가 문제 원인 조사 및 수정	코멘트 : “API 세션 관리 오류 수정 중”
❹ 종료	수정 작업 후 리뷰 완료	PM : 테스트 완료 후 Issue closed

❖ 실제 Issue 관리 사례

 **lokithoth** added **dotnet** **feature** **proj-extensions** on Oct 24, 2024

 **lokithoth** added this to the **0.4** milestone on Oct 24, 2024

 **rysweet** assigned **LittleLittleCloud** on Oct 25, 2024

 **rysweet** added **size-medium** **size-large** and removed **needs-triage** on Oct 25, 2024

효과적인 Issue 관리 방법

❖ 좋은 Issue를 작성하는 방법

1. 명확한 제목 작성

- 이슈의 제목만으로도 내용을 이해할 수 있게 작성
- 문제의 핵심을 짧고 명확하게 표현
- 기술적 맥락을 포함하여 이해하기 쉽게 작성

2. 상세한 문제 생성 및 재현

- 문제를 재현할 수 있는 방법을 상세히 기술
 - 로그인 화면 진입
 - 로그인 버튼 클릭
 - 오류 메시지 발생

3. 스크린샷 및 증거 첨부

- 문제 상황을 보여주는 스크린샷 첨부

4. 기대 결과 포함

- 문제가 해결되었을 때의 기대 결과를 명확히 기술

UI Bug: Team Builder/Playground Blank in Chrome - Resolved by Hard Refresh proj-studio

Bug #5943 · usag1e opened 2 days ago

- 버그 유형(UI Bug) + 원인(Chrome에서 빈 화면) + 해결 방법(Hard Refresh로 해결됨)까지 포함
- 문제를 빠르게 이해하고 재현할 수 있도록 구체적인 정보 제공

Create a sample showing how to set up OTEL and see results in Jaeger UI help wanted sample-request

Feature #5892 · ekzhu opened 5 days ago · 0.4.x-python

- OTEL 설정 및 Jaeger UI에서 결과 확인 방법을 보여주는 샘플 요청을 명확히 전달
- "Create a sample"이라는 명확한 작업 요청 포함

usag1e opened 2 days ago · edited by usag1e

What happened?

Describe the bug

Issue 설명

- AutoGen Studio UI (Team Builder and Playground) initially fails to load in Google Chrome, displaying blank sections.
- The Gallery section loads correctly.
- A hard refresh (`Ctrl+Shift+R`) in Google Chrome resolves the issue. I guess this is a caching problem. The UI functions correctly in Microsoft Edge without requiring a hard refresh.
- Additionnaly, with version `>=0.4.2.dev`, after the hard refresh, there are issues on both Edge and Chrome when clicking on `Team Builder` or `New team` for example (see error below)

To Reproduce the caching problem

I tried with AGS version: 0.4.1.11 to 0.4.2.dev3

Default Browser: Google Chrome (Version 134.0.6998.89 (Official Build) (64-bit)).

재현 방법

- Team Builder and Playground are blank. Gallery loads.
- Hard refresh (`Ctrl+Shift+R`): Team Builder and Playground should load.
- Try creating a new team.

Screenshots versions from 0.4.1.11 to 0.4.2.dev3 with Chrome without hard refresh

- Playground (same for Team Builder)

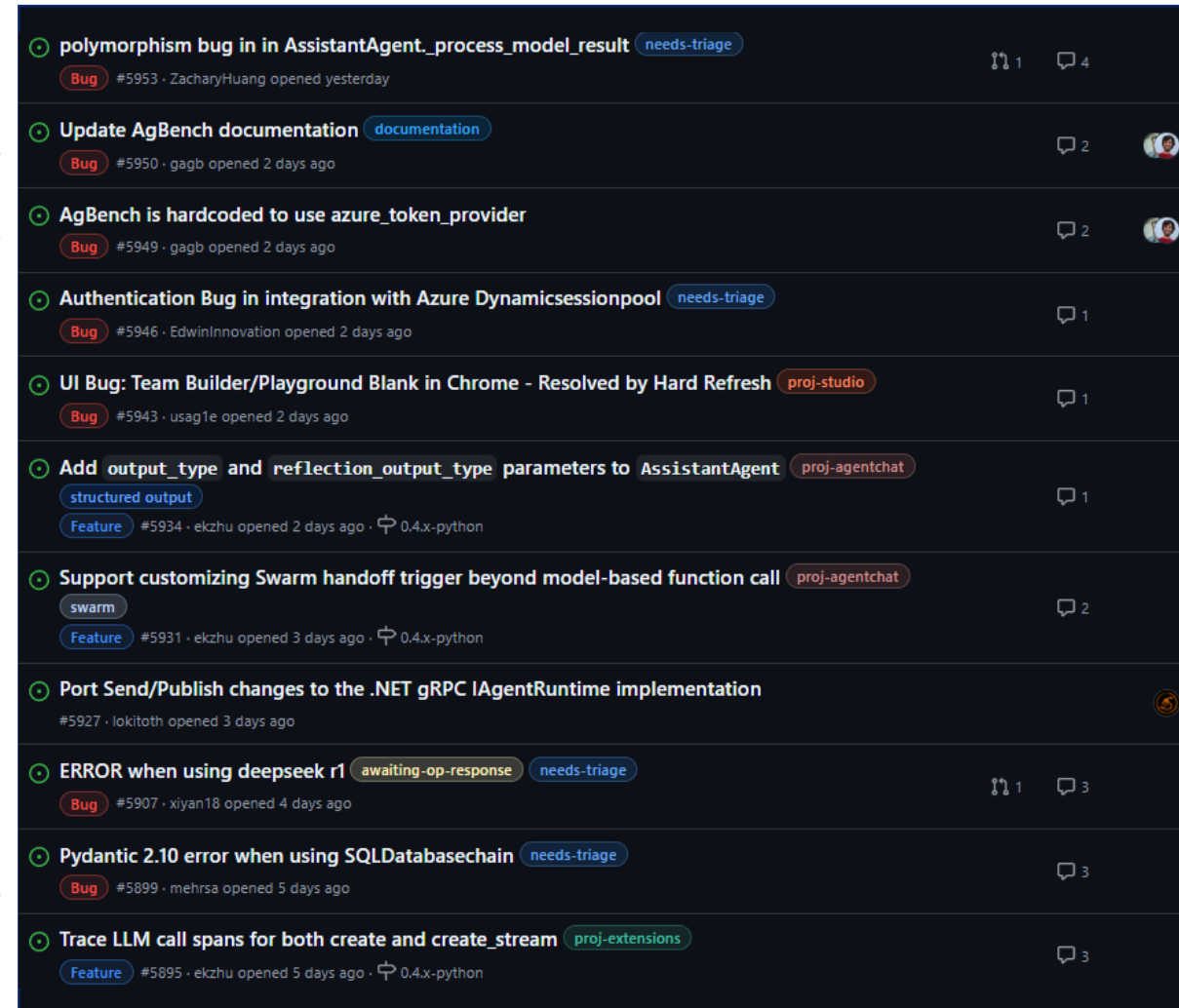
Expected behavior

All UI sections (Team Builder, Playground, and Gallery) should load correctly in Google Chrome on the initial load, without requiring a hard refresh. Creating new teams and interacting with the UI should function without errors.

효과적인 Issue 관리 방법

❖ Issue 관리 시 지켜야 할 원칙

기준	권장 방법	기대 효과
명확한 제목	<ul style="list-style-type: none"> 구체적인 제목 작성 업무를 정확히 Assignee에게 할당하여 책임 소재를 명료하게 규정함 	업무 혼란 방지, 빠른 대응 가능
상세한 설명	<ul style="list-style-type: none"> 환경, 재현 방법을 상세하게 작성 	문제 원인 신속히 파악 가능
적절한 Labeling	<ul style="list-style-type: none"> Issue 유형과 중요도를 명확히 표시하는 라벨 활용 업무 우선순위 관리 	업무 효율성과 대응 속도 향상
담당자 지정 (Assignee)	<ul style="list-style-type: none"> 업무를 정확히 담당자에게 할당하여 책임 소재 명확화 	책임감 증가, 업무 지연 방지
상태 업데이트	<ul style="list-style-type: none"> 진행 상태 지속적 업데이트 Issue 진행 과정을 주기적으로 업데이트하여 공유 	업무 진행 현황 실시간 공유



▲ 활발한 Issue 관리 화면 예시

오픈소스 프로젝트의 Issue 활용 사례

사례 1: PyTorch 프로젝트의 Issue 관리



pytorch

Verified

10.6k followers

where the eigens are valued

<https://pytorch.org>

❖ PyTorch 의 Issue 관리 특징

1. 철저한 Label 관리
 - 이슈 유형(버그, 기능 추가 등), 중요도, 긴급성을 명확히 구분하여 관리
2. 명확한 담당자 할당
 - 각 Issue마다 관련 개발자 및 관리자에게
 - 명확히 할당하여 책임 소재 명확화
3. 장기적인 Issue 관리 미팅
 - 프로젝트 관리자들이 주기적으로
 - Issue를 검토하고 우선순위를 설정

❖ 실제 PyTorch 에서 발생하는 Issue 예시

General MPS op coverage tracking issue feature module: mps tracker triaged 2 1715
#77764 · albanD opened on May 19, 2022

Feature Request & Tracking : MPS 백엔드 연산자 지원 확대

- PyTorch의 MPS(Metal Performance Shaders) 백엔드에서 미지원 연산자(Ops) 추적 및 추가 요청
- 더 많은 연산자를 MPS에서 직접 실행하여 성능 향상 및 CPU fallback 감소

Python/C++ API Parity: torch.nn modules and functional good first issue module: cpp module: nn triaged 109
#25883 · yf225 opened on Sep 10, 2019

Feature Request & Tracking : Python/C++ API 기능 일치 (API Parity)

- PyTorch의 Python API(torch.nn)와 C++ API(torch::nn) 기능을 동일하게 확장
- C++에서도 Python API와 동일한 신경망 계층 및 활성화 함수 제공
- 고성능 애플리케이션 개발 및 배포 지원

오픈소스 프로젝트의 Issue 활용 사례

사례 2: Hugging Face 프로젝트의 Issue 관리



Hugging Face

The AI community building the future.

Verified

46.2k followers

NYC + Paris

<https://huggingface.co/>

@huggingface

❖ Hugging Face 의 Issue 관리 특징

1. 커뮤니티 중심의 열린 소통
 - 모든 사용자가 자유롭게 질문하고 이슈 등록 가능
 - 관리자는 적극적으로 답변 제공
2. Issue 템플릿 적극 활용
 - Issue 생성 시 자동 템플릿을 제공
 - 사용자들이 필요한 정보를 기재하도록 유도
3. 주기 정기 관리 시스템
 - 관리자가 미해결 Issue를 점검하여 신속히 처리

❖ 실제 Hugging Face 에서 발생하는 Issue 예시

Community contribution: Adding Flash Attention 2 support for more architectures Good Second Issue

1 110

#26350 · younesbelkada opened on Sep 23, 2023

Feature Request : Flash Attention 2 지원 확대

- Transformer 모델의 속도 향상 및 메모리 최적화를 위해 Flash Attention 2 지원 확대
- 각 모델별 Flash Attention 2 모듈 구현 후 PR 제출
- LLM 학습 및 추론 성능 개선

Community contribution: Adding GGUF support for more architectures

Feature request Good Second Issue

32

#33260 · SunMarc opened on Sep 2, 2024

Feature Request : GGUF 지원 확장

- Transformer 모델을 저사양 환경에서도 효율적으로 실행할 수 있도록 GGUF 지원 확대
- GGUF 텐서 매핑 및 테스트 코드 작성 후 PR 제출
- 모델 실행 속도 개선 및 메모리 사용 최적화

Issue 관리 문제와 대응 전략

문제 1 : Duplicate Issue (이슈 중복 발생)

▶ 문제 정의 및 원인

- 이미 등록된 문제를 다른 사용자가 중복 등록하여 Issue 난립 현상
- 기존 Issue 검색 부족 or Issue 확인 어려운 구조일 경우

▶ 부정적 영향

- 중복된 Issue 관리로 인한 관리 비용 증가 및 혼란 초래
- 실제 중요한 Issue 처리 지연 및 프로젝트 효율성 저하

▶ 효과적인 대응 방법

- 명확한 Issue 템플릿 제공
- 관리자가 주기적으로 중복 Issue 검토, 병합, close

Duplicate Issue 처리 예시

Issue #245: "로그인 기능 속도 개선"

Closed 이 Issue는 이미 #125에서 다루고 있습니다.

문제 2 : Stale Issues (방치된 이슈 증가)

▶ 문제 정의 및 원인

- Issue에 대한 우선순위 설정이 제대로 되지 않은 경우
- 담당자 할당이 제대로 되지 않은 경우

▶ 효과적인 대응 방법

- 장기적인 Issue 관리 회의를 통해 미처리 Issue 이슈 점검 후 우선순위 재조정
- 일정 기간 이상 미처리된 Issue 자동으로 알림을 보내거나 상태 재점검

Stale Issue 관리 예시

Issue #112: "문서 업데이트 요청"

Sale 30일 이상 업데이트 없음

[BOT] 이 Issue는 30일 이상 활동이 없어 Stale로 표시

Issue 관리 문제와 대응 전략

문제 3 : 우선 순위 혼란으로 인한 Issue 문제

▶ 문제 정의 및 원인

- Issue의 우선 순위가 정확히 설정되지 않아 중요하지 않은 Issue가 먼저 처리되는 경우가 발생

▶ 효과적인 대응 방법

- Labeling 체계를 도입 및 철저한 적용 (High Priority, Low Priority 등)
- 주기적 우선순위 평가 회의를 통해 및 중요도를 지속적으로 관리

문제 4: 불충분한 정보로 인한 Issue 문제

▶ 문제 정의 및 원인

- 작성된 Issue가 명확하지 않거나 정보가 불충분하여 문제 파악 및 대응이 어려운 경우 발생

▶ 효과적인 대응 방법

- Issue 템플릿을 명확하게 구성하여 필수 정보
- 환경, 재현 방법, 로그 등을 기재하도록 유도
- Issue 작성 시 필요한 정보를 정확히 안내하는 가이드 제공

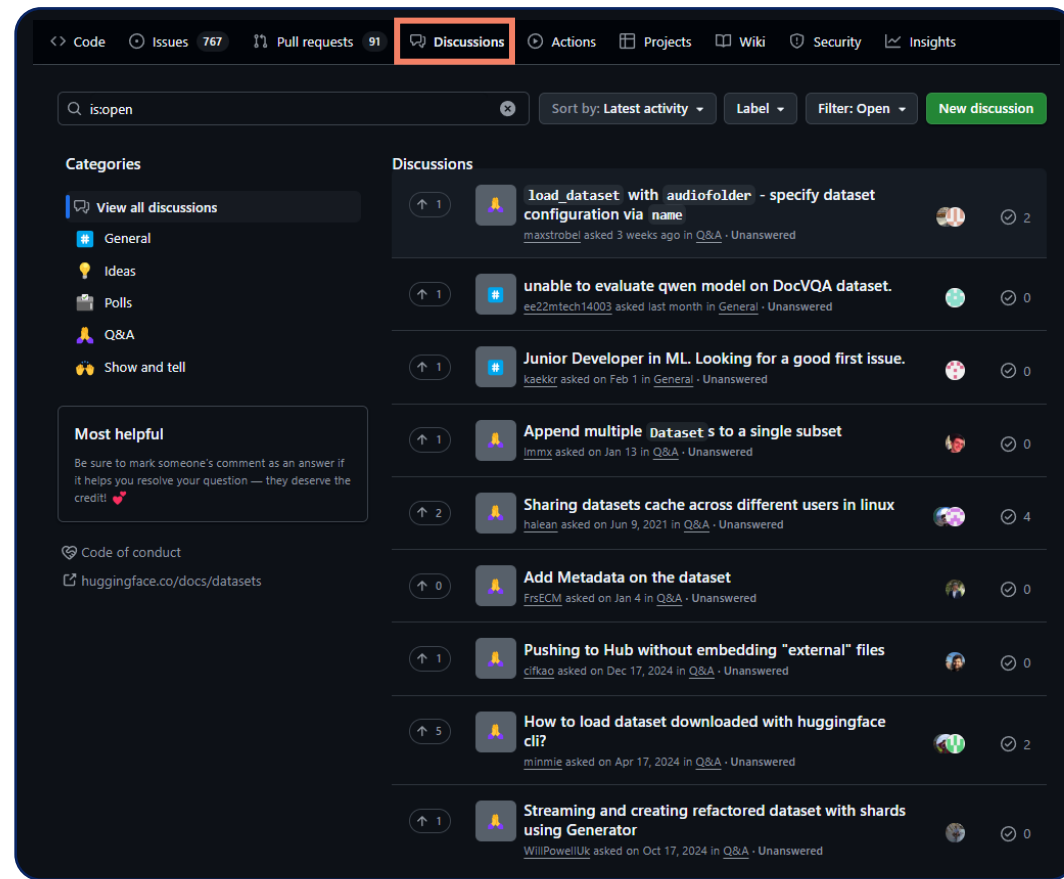
GitHub Discussions

GitHub Discussions

- 프로젝트 커뮤니티 내에서 소통을 위한 공식적인 토론 공간
- 유지 관리자, 기여자 및 방문자가 중앙 집중식으로 논의 가능
- 질문, 아이디어 제안, 피드백 수집 등 의견 교환과 커뮤니티 형성이 주요 목적

❖ 주요 활용 방법

- 공지 사항 및 정보 공유
- 피드백 수집 및 계획 논의
- 질문 및 답변
- 커뮤니티 의견 수집(설문 조사)
- 찬성(Upvote) 기능으로 아이디어 가시성 향상
- 커뮤니티 분위기 조성



▲ GitHub Discussions 예시

GitHub Discussions

❖ Issues와 Discussions의 차이점

구분	Issues	Discussions
목적	작업(Task), 버그(Bug) 관리 및 문제 해결	아이디어 논의, Q&A, 커뮤니티 소통
초점	명확한 업무 단위 및 실행 중심	열린 대화 및 의견 교환 중심
진행 방식	문제 해결 후 Close(완료)	지속적인 논의 가능
활용 사례	특정 버그 수정, 기능 구현 요청	기능 제안, 프로젝트 방향 논의, 질의응답(Q&A)

❖ Issues와 Discussions의 활용 예시

Issues가 적합한 경우

- 해결이 필요한 구체적인 문제가 있을 때
- 업무(Task)나 버그(Bug) 수정이 필요할 때
- 해결이 완료되면 Close 가능

Discussions가 적합한 경우

- 장기적이고 개방적인 논의가 필요할 때
- 특정 문제 해결보다는 아이디어나 피드백을 공유할 때
- 다양한 의견을 지속적으로 수렴할 때

05 GitHub Issues와 Notion의 연계 관리

프로젝트 관리 도구 연계

❖ 프로젝트 관리 도구 연계의 필요성

1. 각 도구의 강점 극대화

- 각 관리 도구는 특정 업무에 특화되어 있음
 - GitHub Issues → 코드 중심의 버그 리포트 및 이슈 관리
 - Notion → 문서 관리, 로드맵 작성, 데이터베이스 통합 관리

2. 정보 관리 효율성 향상

- 여러 도구에서 개별적으로 정보를 관리하면 중복 업무 및 정보 분산 발생
- 자동 동기화 및 공유로 업무 관리 효율 극대화

3. 협업과 업무 투명성 강화

- 팀원들이 동일한 정보를 실시간으로 공유
- 정보의 투명성 증가 및 프로젝트 생산성 향상

❖ 프로젝트 관리 도구 연계를 통해 얻을 수 있는 이점

1. 정보 집중화 (Information Centralization)

- 분산된 정보를 Notion에서 한 번에 확인하여 업무의 효율성 극대화

2. 실시간 정보 공유 (Real-time Information Sharing)

- GitHub Issues의 정보가 실시간으로 Notion에 동기화
- 팀 전체가 최신 정보를 즉각적으로 공유

3. 프로젝트 가시성 증가 (Enhanced Project Visibility)

- 프로젝트의 전체 현황과 세부 업무의 상태를 Notion에 기재
- 업무 진행 상황과 병목 지점을 빠르게 발견하여 프로젝트 관리 개선 가능



Notion



Jira



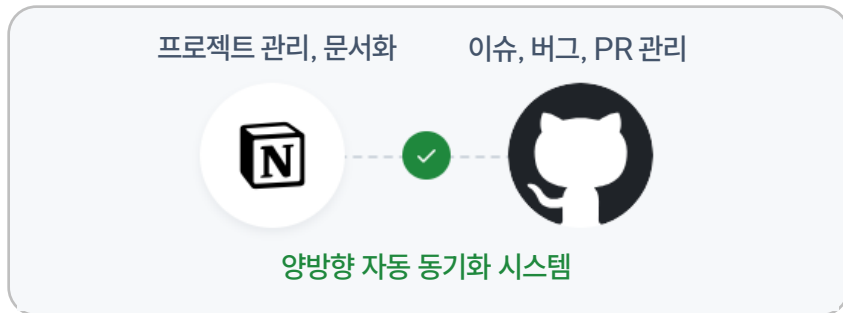
slack



JANDI

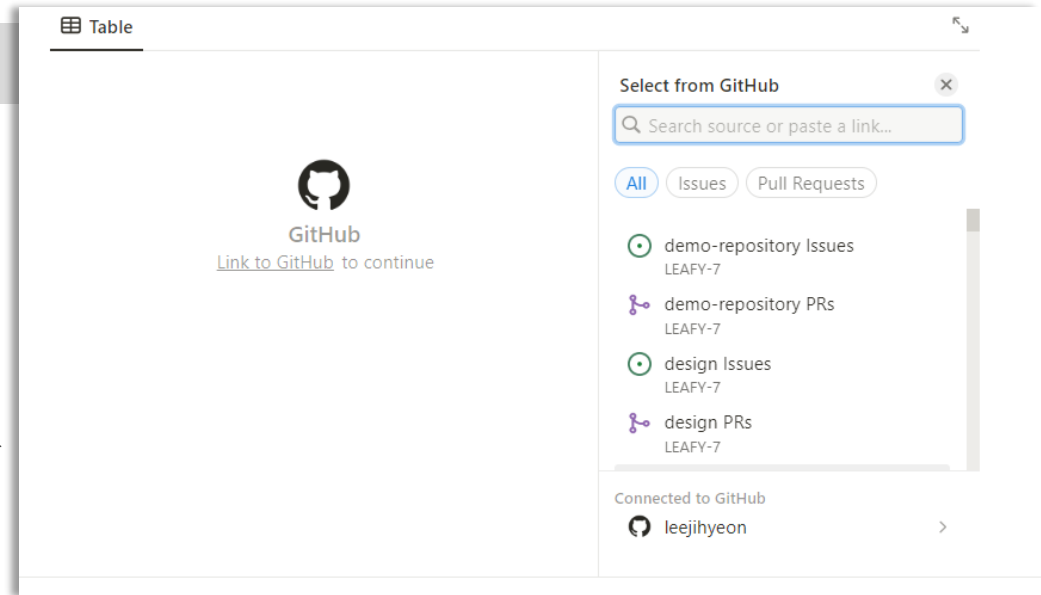
프로젝트 관리 도구 연계 : GitHub Issues ↔ Notion

❖ GitHub Issues ↔ Notion 연계



▲ GitHub와 Notion의 연계 개념도

Notion의 Database에 ▶
GitHub를 연동하는 화면



❖ GitHub Issues ↔ Notion 연계 시 고려해야 할 이론적 요소

1. Avoiding Duplicate Management

- 문제 사항
 - GitHub와 Notion에서 동일 정보의 이중 관리로 혼선 및 업무 복잡성 증가
- 이론적 대응 방안
 - 단일 정보 원칙(Single Source of Truth)
 - 원본(Source) 하나로 통일
 - 나머지는 참조 방식 운영

2. Synchronization Management

- 문제 사항
 - 비동기, 지연 발생시 정보 불일치로 업무 혼란 초래 가능
- 이론적 대응 방안
 - 자동화 도구(GitHub Actions, Zapier)
 - 실시간 또는 빈번한 동기화로 정보 최신화 유지

3. Clear Guidelines

- 문제 사항
 - 복잡한 연계로 팀원 혼란 및 잘못된 정보 입력 가능
- 이론적 대응 방안
 - 명확한 가이드 제공(이슈 작성, Labeling 규칙)
 - 정보 흐름 일관성 유지

프로젝트 관리 도구 연계 - 사례

❖ GitHub Issues ↔ Notion 연계 흐름

단계	흐름
① Issues 생성	<ul style="list-style-type: none"> GitHub에서 사용자가 새로운 이슈 생성
② 실시간 동기화	<ul style="list-style-type: none"> GitHub Actions와 같은 자동화 도구를 통해 이슈가 즉각적으로 Notion 데이터베이스에 동기화
③ Notion을 통한 업무 관리	<ul style="list-style-type: none"> 동기화된 이슈가 Notion에서 개발자와 비개발자 간에 실시간으로 공유되며, 우선순위와 처리 현황도 추가적으로 함께 관리
④ GitHub로 상태 업데이트 재동기화	<ul style="list-style-type: none"> Notion으로 업무 상태 업데이트시, GitHub Issues로 실시간 동기화 개발 관련 작업과 문서화 작업을 한 곳에서 관리함으로써, 업무의 연속성이 향상

❖ 사례) Heritage Holdings Tech (미국, 투자 플랫폼)



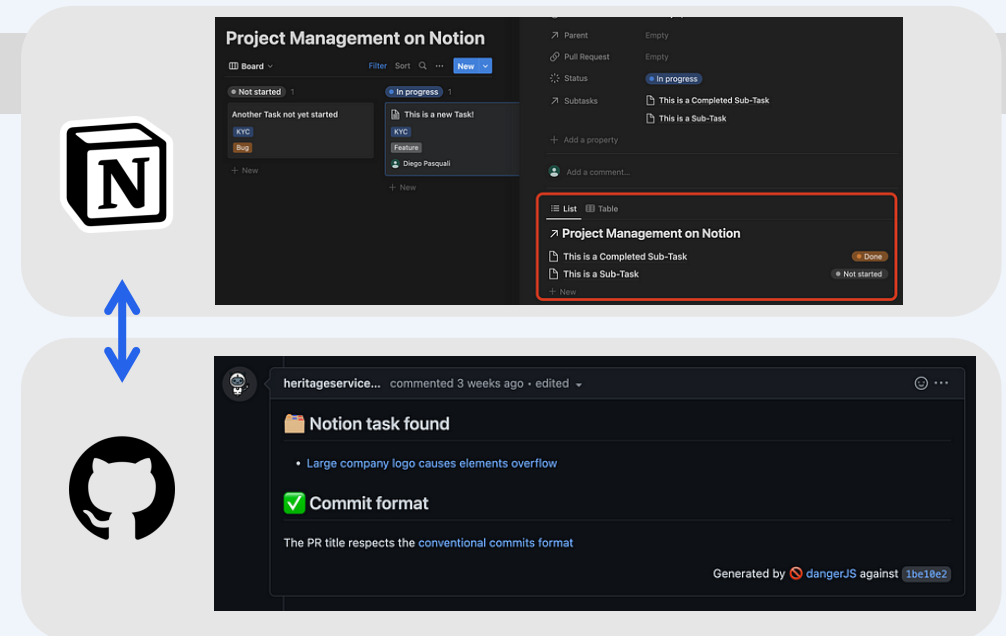
- Notion을 주요 프로젝트 관리 도구로 사용하고, GitHub와의 연계를 통해 자동화된 task 관리 시스템을 구축 → 중복 관리 문제를 해결하고 업무 진행 상황을 실시간으로 업데이트

Notion을 프로젝트 관리의 중심 도구로 활용

- 중앙 지식 베이스 역할 수행
- 여러 프로젝트에서 태스크를 체계적으로 관리

GitHub과 연계하여 태스크 진행 상황 자동 업데이트

- Pull Request(PR) 상태 변경에 따라 Notion의 task 상태 자동 변경
- PR과 연계된 태스크 정보 자동 기록



프로젝트 관리 도구 연계 – 도구 선택 시 고려사항

❖ 프로젝트 관리 도구 선택 시 고려사항

① Accessibility & Usability

- 모든 참여자가 쉽게 접근 가능
- 팀원의 기술과 경험 수준에 맞는 도구 선택이 필요

▶ 고려해야 할 사항

- 신규, 비기술적 멤버도 쉽게 적응 가능한 인터페이스 제공이 중요
- 학습 부담이 적은 도구 선택을 권장

② Integration & Compatibility

- 관리 도구 간 연동 및 정보 공유 가능 여부 확인
- 효율적 정보 흐름을 위해 중복 방지와 도구 연계 필수

▶ 고려해야 할 사항

- GitHub Issues와 Notion 등 주요 도구의 연계 가능성 확인
- Zapier, Automate.io 등 자동화 솔루션 활용 가능성 검토

③ Scalability & Flexibility

- 프로젝트 확장이나 조직 성장 시 도구의 확장 가능성 검토
- 변화하는 요구 사항에 대한 적응 및 확장 가능성 평가

▶ 구체적인 예시 상황

- 작은 팀이 Notion과 GitHub로 시작해도 성장 후 확장 용이
- 확장성이 낮으면 관리 어려움과 생산성 저하 초래

④ Clear Objectives

- 프로젝트의 목표가 명확해야 팀원들이 업무를 정확히 이해 가능
- 목표 설정이 모호한 경우 : 업무와 우선순위 혼란 유발

▶ 명백한 역할과 책임 설정

- 각 팀원의 역할과 책임을 분명히 정해 업무 책임감과 참여도 높임
- 업무 중복 및 책임 회피를 방지해 관리 효율성 향상

06 마무리 및 Q&A

요약

◆ 프로젝트 팀 구성과 역할 분담의 중요성 강조

- 팀 구성은 단순히 사람을 모으는 것이 아니라, 각 팀원의 역할과 책임을 명확하게 정의하고 협력을 촉진하는 기반을 마련하는 과정
- 좋은 역할 분담은 팀 내 커뮤니케이션과 협업을 증진시키고, 프로젝트 목표 달성에 결정적인 역할
- 역할의 명확성, 업무 책임의 명확한 분배를 통해 책임 회피와 업무 중복 예방

◆ 프로젝트 관리 도구의 필요성 및 연계 이론

- 프로젝트 관리 도구는 업무를 시각화하고 명확하게 정의함으로써 효율성을 높임
- 특히 GitHub Issues는 이슈 단위로 업무를 정의하여 업무의 명확성과 투명성을 극대화
- Notion은 다양한 업무 정보를 중앙에서 체계적으로 관리할 수 있도록 지원
- 두 도구 간의 연계를 통해 정보의 중앙화와 실시간 정보 공유를 가능하게 하여 협업 효율성 극대화

◆ 프로젝트 관리 도구 선택과 활용의 Best Practice 강조

- 팀의 규모와 특성, 프로젝트의 목표에 맞춰 적합한 도구 선택이 필수적
- 이슈 관리 시 명확한 제목, 상세한 설명, 적절한 labeling, 명확한 담당자 지정이 필수
- 도구 간 연계를 통해 정보의 중앙화를 이루고, 관리 효율성을 높이는 것이 중요

다음 실습 안내

- ▶ Notion 기초 (Page, Kanban, Database)
- ▶ Notion과 GitHub Issues 연계

