Software V&V
Final Presentation

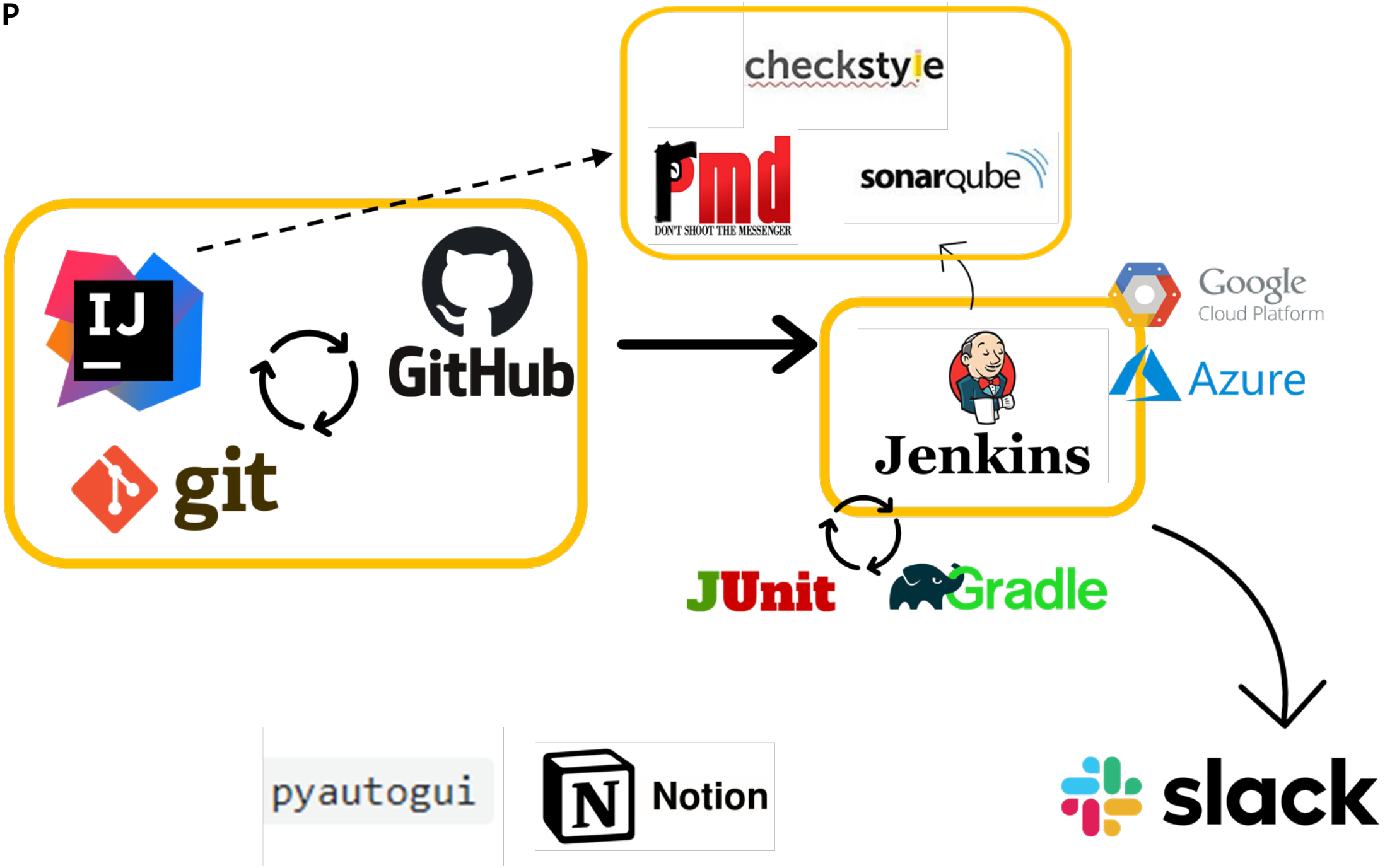
※ Contents

1. CTIP
2. Summary / **A1**, **A2**, **B1**
   - SQA Activities
   - Comments

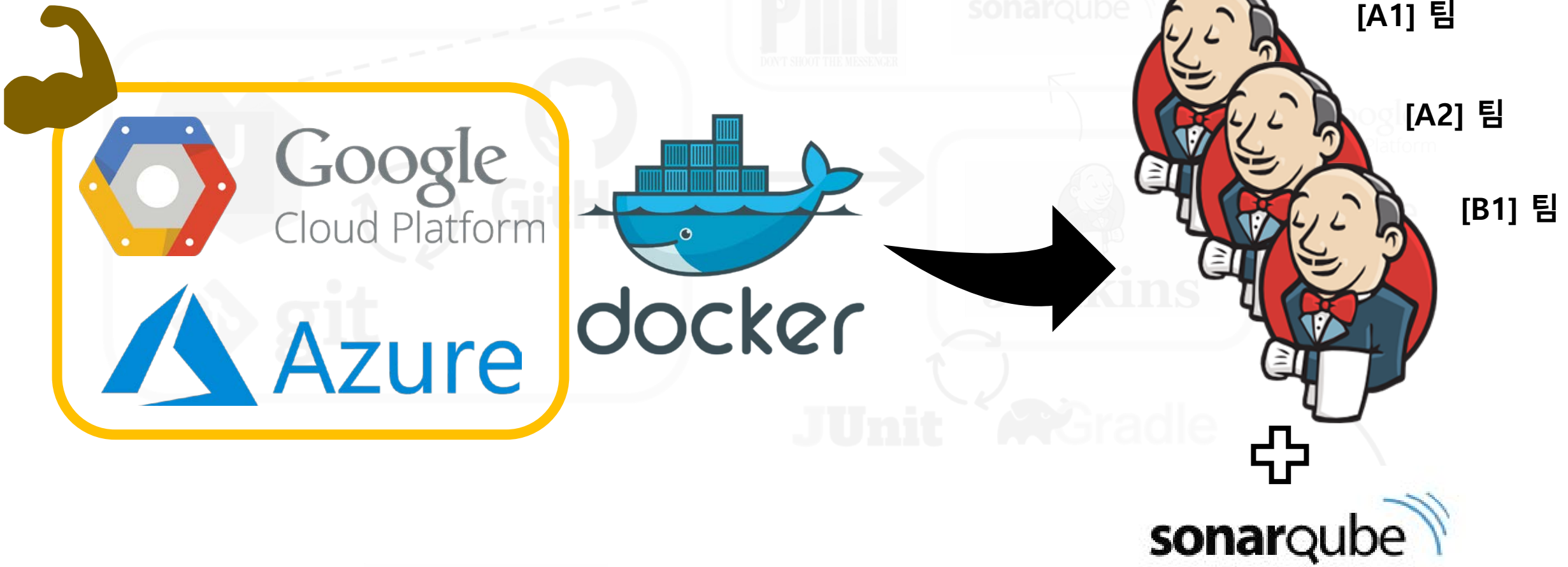
Team #1
201411273 박재범
201411295 이상훈
201510436 허윤아
201511244 김민우

Dependable Software Laboratory

1. CTIP

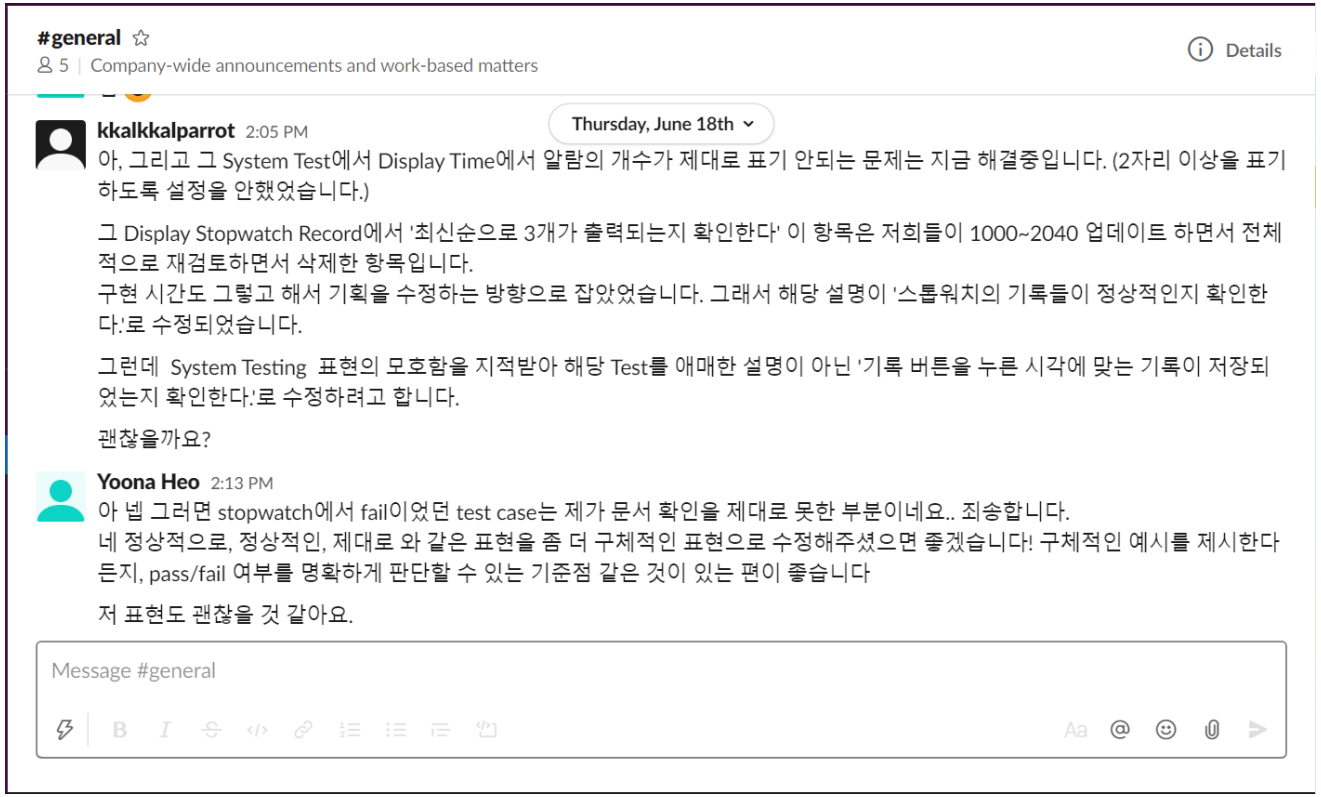

checkstyle
pmd
DON'T SHOOT THE MESSENGER
sonarqube
IJ
GitHub
git
Google
Cloud Platform
Azure
Jenkins
JUnit
Gradle
pyautogui
Notion
slack

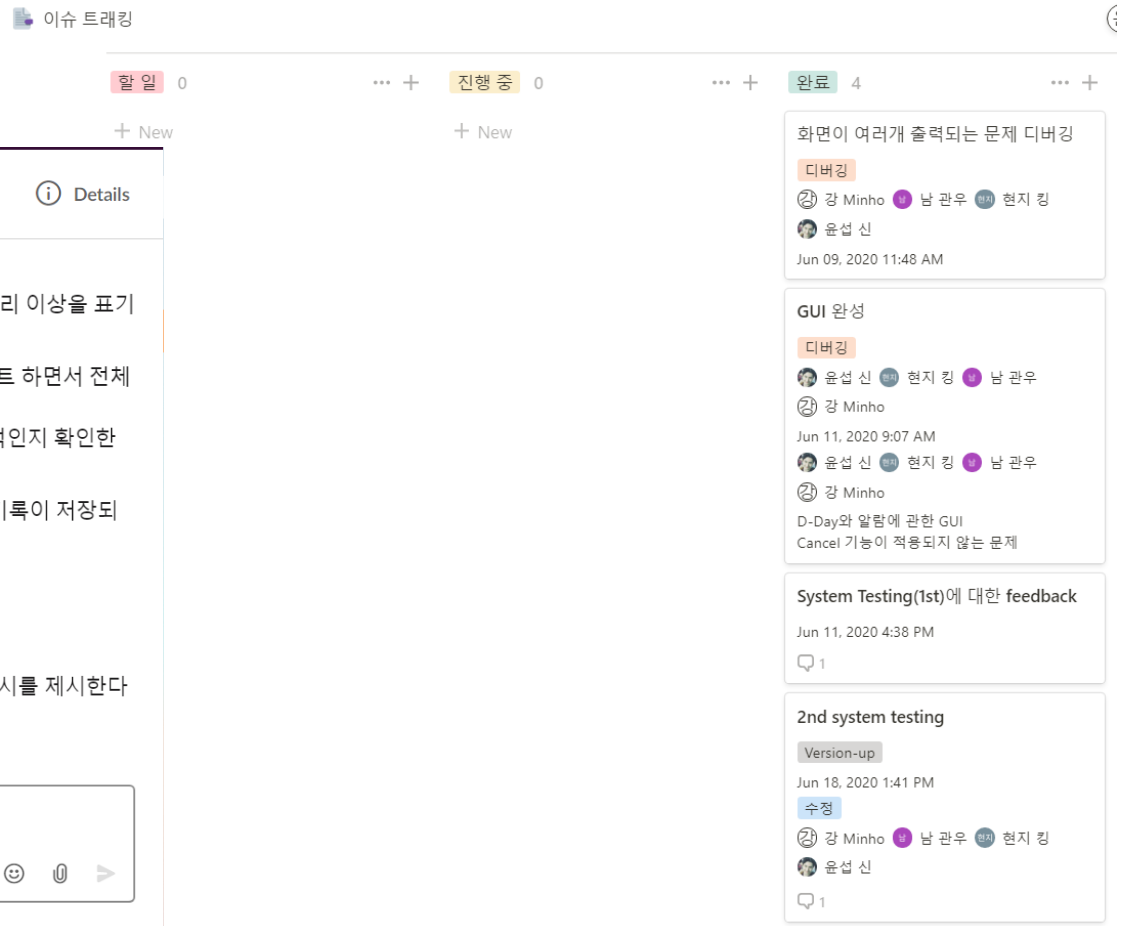
1. CTIP



2. Summary / [A1] 허윤아 / SQA Activities

- CTIP 환경 구축 시 계속해서 문제가 발생해서, 많은 기여를 하지는 못했음
- 개발팀과 Slack으로 실시간 소통(github, Jenkins, notion을 연동)
- System testing을 진행하면서 발생한 Issue는 Notion에 등록





2. Summary / [A1] 허윤아 / Comments

힘들었던 점

- 초반에 기껏 구축한 서버가 갑자기 먹통이 되는 등 지속적인 실패, 이로 인한 자신감의 하락
- 팀원들에게 짐만 되는 것 같다는 죄책감

좋았던 점

- 좋은 팀원들
- 3학년 학생들과의 원활한 소통과, 그를 통해 얻어낸 좋은 결과물
- 많은 것을 배울 수 있었던 실습 그 자체

좋은 개발팀 & 좋은 SQA

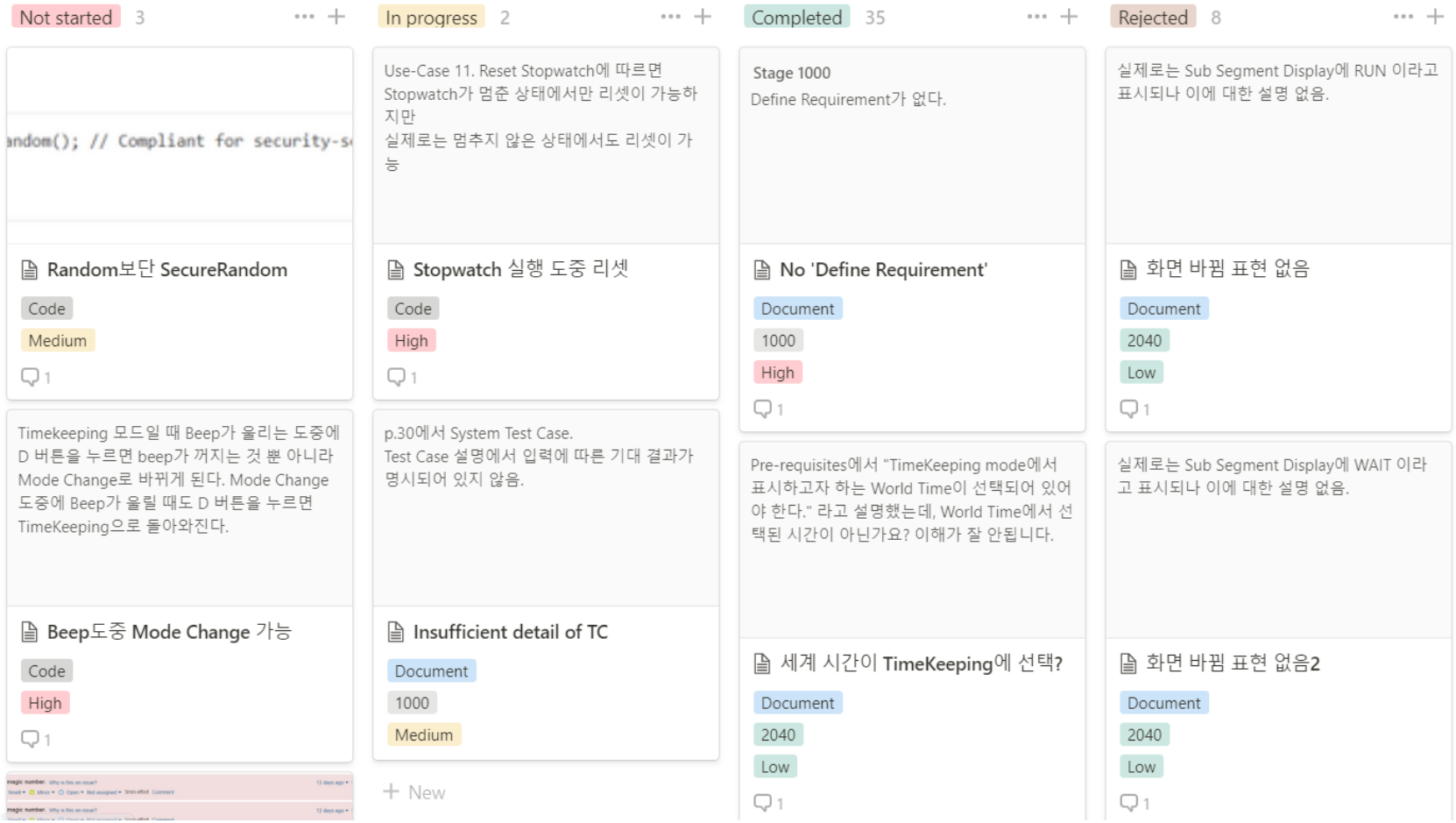
- 소통을 할 줄 알고, 협업의 중요성을 알고, 공사 구분이 철저한 사람들

총평

- SQA의 주관으로 판단하는 부분이 정말 많기 때문에, SQA의 역량이 중요하다는 것을 실습을 통해 느꼈음
- 정적 분석 도구에서 정리된 rule set을 적용했음에도 불구하고 봐야 할 내용이 정말 많았음
  이를 통해 우리가 개발을 할 때 지켜야 할 rule 하나하나가 중요하다는 것을 깨닫게 되었음
- 3학년 때 소프트웨어 모델링 수업을 들을 당시 우리 팀은 나쁘지 않은 개발팀이라고 생각했는데, 담당했던 4학년 팀의 마음을 이제라도 깨닫게 되었음

2. Summary / [A2] 이상훈 / SQA Activities

- CTIP 환경 구축.
- 개발팀과 Microsoft Team로 소통. 개발 간 궁금한 사항을 실시간으로 묻고 답하는 활동.
- Spec Review, System Test와 Static Analysis 결과로 발생한 Issue를 Notion에 등록.



2. Summary / [A2] 이상훈 / Comments

짧은 소감

- CTIP 도구가 굉장히 많다.
- 오픈소스 도구로 CTIP 환경 구축하기가 까다로웠다. 유료 도구도 써보고 싶다.
- 개발 초기부터 개발팀과의 원활한 소통이 중요하다.
- 소프트웨어 개발에서 Spec 문서의 중요성이 상기됐다.
- SQA의 주관이 은근히 많이 들어간다.
- Category Partitioning Test나 Brute Force Test 같은 System Test는 온전히 SQA 능력에 달렸다.
- Pair wise test tool은 요술봉 같다.
- 정적 분석 도구는 앞으로 계속 사용하고 싶다.
- 우리가 지켜야 할 규칙이나 코딩 스타일이 이렇게 많은 줄 처음 알았다.

2. Summary / [A2] 이상훈 / Comments

힘들었던 점

- 오래된 오픈소스 도구를 쓰기가 너무 까다로웠다.
- System Test를 모두 돌려보는 작업이 고된 노동이었다.

좋은 SQA

- 개발팀과 원활한 소통.
- 개발팀의 일원처럼 활동하되 제3 자의 시각으로 검증.

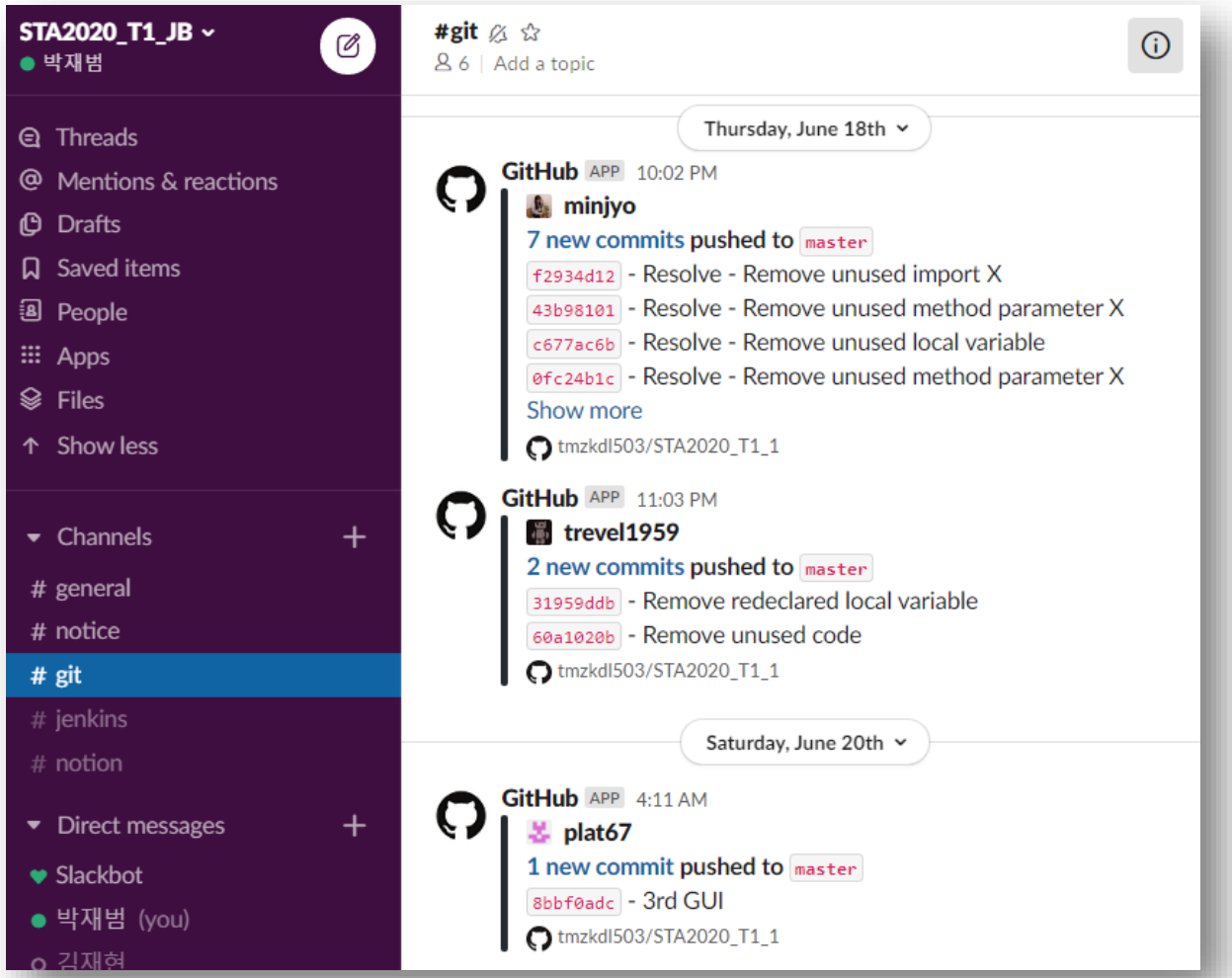
나쁜 SQA

- 개발팀과 소통되지 않고 독단적인 SQA
- 개발팀과 온전히 하나되어 객관적인 검증이 되지 않는 SQA

좋은 개발자

- 소프트웨어 개발은 협업이라는 걸 안다.
- 개발 뿐 아니라 CTIP 환경에도 관심을 기울인다.

2. Summary / [B1] 박재범, 김민우 / SQA Activities



◈ Slack을 통한 커뮤니케이션

용도별 5개의 Channels 운영

- **General :** 자유롭게 의견 교환
- **Notice :** 공지사항(SQA to Developers)
- **Git :** 깃허브 푸시 내역 확인
- **Jenkins :** 젠킨스 업데이트 내역 확인
- **Notion :** 노션 업데이트 내역 확인

◈ Notion을 통한 이슈 트래킹

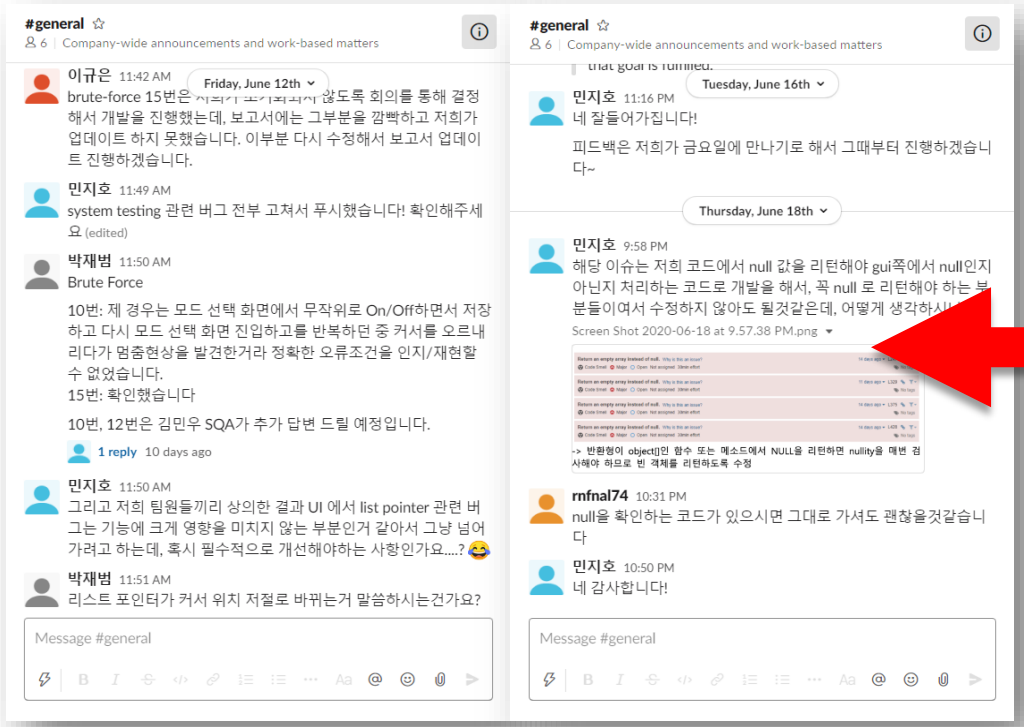
이슈, 작업종류, 우선순위, 담당자, 기한 등

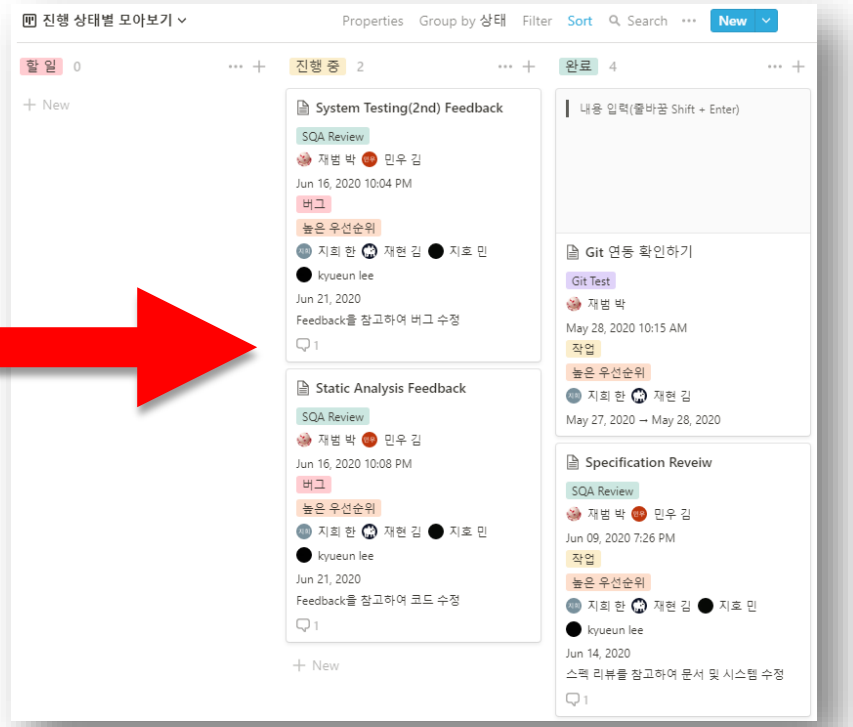
2. Summary / [B1] 박재범, 김민우 / SQA Activities

◈ 커뮤니케이션 프로세스

V&V 프로세스 수행 → Notion에 이슈 등록 → Slack Notice 채널에 공지

→ Slack General 채널로 질문 및 피드백 → SQA가 완료된 이슈를 Confirm 후 종료





slack

Notion

2. Summary / [B1] 박재범 / Comments

1. CTIP(Continuous Test & Integration Platform) 활용

초기 환경 설정(서버 구축 및 설정, 플러그인 설치, Tool Integration)이 가장 어려웠다.
CTIP 구축 시 주안점은 **"요즘 실무에서 많이 사용되는 진입장벽이 낮은 도구들를 사용해보자", "도구 자체의 전문적인 사용보다는 CTIP을 활용한 전체 프로세스의 경험(SQA, 개발자 모두)에 중점을 두자"** → **도구는 시간이 지나면 대체되지만 큰 흐름을 이해하고 있으면 응용이 가능하다.**

▶ **생각했던 방향성에 맞는 유익한 경험이 되었다.**

2. Issue Tracking에 대한 생각

이슈(에러, 버그, 스펙 리뷰 등) 관리 측면에서 어떤 게 더 효율적인가?
항목 하나 하나를 개별로 등록하는 것 vs 카테고리 단위로 묶어서 등록하는 것
이슈 자동 등록 시스템이 없는 환경이라면 Category Partitioning Testing처럼 경우에 따라 100개 이상의 이슈가 발생하는 경우 일일이 등록하는 것은 번거롭고 비효율적인 것 같다.

▶ **장단점이 있는 것 같다.**

2. Summary / [B1] 박재범 / Comments

3. 한 학기로 끝나는 수업의 일회성 프로젝트라는 한계

수업 특성상 **CTIP환경 및 코드의 재활용, 유지보수 프로세스**까지 프로젝트가 지속될 수 없어서 **CTIP의 효용성**을 체감하기 어려웠고 일반적인 개발 프로세스에 비해 효율적이라는 느낌이 들지는 않았다. 또한 SQA, 개발자 모두 다른 수업과 병행해서 진행하기 때문에 집중도가 분산될 수 밖에 없어서 아쉬웠다.

4. 아쉬웠던 점

- 코로나 바이러스 때문에 100% 온라인 수업으로 진행됐고 개발 일정도 미뤄져서 **각 프로세스를 진행할 때 시간적으로 여유롭지 못했던 게** 아쉽다.
- 팀프로젝트였지만 개인별로 개발팀을 맡게 되어서 같은 팀 내 SQA와 의견을 주고받을 일이 거의 없었고 개인 프로젝트처럼 느껴졌다.
- **SQA(4학년)가 개발자(3학년)의 작업물을 평가 및 테스트하는 관계**이다보니 개발자들이 솔직하게 애로사항 등을 얘기하기 어렵고, 이 때문에 개발자들이 SQA의 리뷰에 대해 어떻게 받아들이고 있는지 파악하기 힘든 것 같다.

2. Summary / [B1] 박재범 / Comments

5. 좋았던 점

- 맡은 팀에 V&V 수업을 같이 듣고있는 개발자들이 있어서 의견 공유 등을 편하게 할 수 있었고 **개발자들이 리뷰한 내용에 대해서도 많이 질문하고 열심히 대응**해줘서 고마웠다.
- 학기 전체가 온라인으로 진행되었지만 Zoom을 활용한 수업 체계가 잘 갖추어져서 조금 타이트한 일정 외에는 불편한 부분 없이 실습을 진행할 수 있었다.
- 개발자가 **개발하면서 놓치는 부분**들이 있을 수 밖에 없는데 체계적인 SQA 활동을 통해서 그런 부분들을 많이 커버할 수 있다는 것을 체감할 수 있었다.

6. 좋은 SQA란?

이론과 개념에 대한 정확한 이해를 바탕으로 **융통성**을 겸비하고 타인의 의견을 적극적으로 수용하면서도 **명확한 기준을 가지고 평가와 테스팅을 진행**할 수 있다면 **좋은 SQA**라고 생각한다.

7. 좋은 개발자란?

팀 단위 개발에서는 개인의 역량만 믿고 독단적으로 작업하지 않고 불편함과 번거로움을 감수하더라도 팀 내부적으로 **약속한 문법이나 룰을 잘 지키고 의사소통을 잘 해야 좋은 개발자**라고 생각한다.

2. Summary / [B1] 김민우 / Comments

1. 타인이 작성한 문서와 코드에 대한 이해

: 코드를 작성한 근본적인 의도를 알지 못하면 주석이 있어도 코드를 이해하는데 시간이 많이 걸릴 때가 있습니다. V&V수업에서 프로젝트 진행 의도를 파악하는데 문서가 얼마나 중요한지 느꼈습니다.

2. 누군가의 작품이나 코드에 수정을 요구하는 일

: 개발을 해본 입장에서 문서나 코드의 수정을 요구하는 부분이 어려웠습니다.
개발자가 의도한 개발방향과 의도가 제대로 전해지기 위해 문서만으로는 부족하다고 느꼈고,
온라인 커뮤니케이션 툴과 더불어 실제 면대면으로 회의를 한다 던지 하는 부차적인 방법이
QA과정을 유연하게 만들 수 있을 것 같다는 생각입니다.
그리고 QA작업은 주관적인 의견이 들어가는 일이지만, 그렇기 때문에 더욱 객관적인 기준도 필요하다고 느꼈습니다

3. 내가 작성했던 코드에 대해서

4학년이 되는 과정 중에 제출한 과제나 프로젝트 코드들을 교수님들은 어떻게 느끼고 채점하셨을지 궁금한 마음이 있었고, 협업을 위한 코드는 어떤 것인지 생각해보는 계기가 됐습니다.

4. 좋은 개발자, 좋은 SQA

SQA입장에서의 좋은 개발자는 확실한 개발 방향과 기준을 가지고 일관성있는 개발을 하는 사람이며, 개발자 입장에서 좋은 SQA는 정확한 기준으로 의견에 근거를 제시하는 사람이라고 생각합니다.