소프트웨어공학 팀 프로젝트

Project 4

Team. 고라파덕

SoftWare Engineering

01. 사업 제안 배경

따라서, 영상 스트리밍 기능을 개선하여 해당 PAIN POINT를 해결해야 함

02. 문제정의 – Pain Point deep Dive

Pain Point

1. 해상도 조절 불가

문제 상황

- 사용하는 브라우저/디바이스에 따라 지원 해상도의 차이가 존재
- 해상도 조절 기능이 존재하지 않음

User은 해상도 차이를 해소할 방법이 없어 불쾌감을 느낌

VoC

"Am I the only one who feels that
Netflix on chrome is lower on quality then other browsers?"

-reddit, subreddit r/netflix

관련 품질 특성

- ***Compatibility – Co-existance***
  다른 종류의 브라우저에서 동일한 퀄리티의 콘텐츠가 제공되어야 함

2. 네트워크 관리 취약

문제 상황

- 인기 컨텐츠의 경우 실시간 스트리밍을 진행하며,
  이때 시청자 수가 증가함
- 동시접속자 수가 폭등할 경우 네트워크 관리가 취약함

수요가 높은 컨텐츠를 적절하게 제공하지 못하고,
User은 이에 대한 불만을 표출하고 있음

VoC

"FAQ's regarding network errors and streaming content quality skyrocketed during the Tyson-Paul matchup"

-Article, 'What's on Netflix?"

관련 품질 특성

- ***Flexibility - Scalability***
  상황에 따라 리소스를 적절히 분배해야 컨텐츠 품질 저하를 방지할 수 있음
- ***Performance Efficiency - Capacity***
  다수의 동시접속자를 Handling 할 수 있어야 함
- ***Performance Efficiency – Time Behavior***
  네트워크에 문제가 생길 경우 사용자에게 즉각적인 컨텐츠 제공이 어려움

3. 영상 제어 기능 불만족

문제 상황

- 스트리밍 시스템의 밝기 조절 기능이 디바이스와 연동이 안되어 있음
- 자막 복수 언어 선택이 불가능함

User는 영상 시청에 있어 제어 기능의 한계에 대한
불편함을 느끼고 있음

VoC

"넷플릭스 볼 때 언어 여러 개 나왔으면 좋겠는데 그게 불가능해서
크롬 익스텐션 써서 보고 있어요"

- 넷플릭스 사용자

관련 품질 특성

- ***Compatibility – Interoperability***
  여러 종류의 기기와 정보를 공유할 수 있어야 함
- ***Funtional Suitability – Functional Completeness***
  다수의 사용자의 목적을 충족할 수 있어야 함

02. 문제정의 – Solution

Solution

1. 해상도 명시 개선

- 균일한 해상도로 컨텐츠를 제공해야 함
- 각 브라우저마다 보안 정책에 의해
  해상도 균일화에 제약이 걸림

해상도 조절 기능 제공이 불가능할 경우,
사용자가 해상도를 미리 인지할 수 있도록
해상도 상태를 제시해야 함

해상도 명시 기능을 추가하고, 브라우저 간 해상도 통일이
불가능할 경우 미리 안내 문구를 띄움

2. 영상 제어 기능 개선

- 영상 플레이어 자체 밝기 조절 시스템이
  너무 밝다고 느끼는 user가 존재함
- 자막 복수 언어 선택이 불가능함

디바이스 별 밝기 설정을 분리해서 서비스를 제공

복수 언어 자막을 지원하도록 플레이어 UI를 수정함

03. 목표 시스템 정의

What

- 사용자의 브라우저/디바이스 별 UI 격차 체감을 줄여주고, 사용자가 불편 없이 콘텐츠에 집중할 수 있도록 향상된 기능을 제공하는 UI 시스템을 구축함

Who

Primary user : 콘텐츠 시청자

- 영상 재생/제어 기능을 사용하는 최종 사용자
- 화질, 자막, 밝기 등 UI 요소를 직접 조작

Secondary user: 서비스 관리자

- 시스템 안정성 모니터링
- 사용자 피드백 반영 및 개선

주요 기능 정의

기능 1: 화질 명시 및 통일성 강화

- 현재 재생중인 콘텐츠 화질을 UI에 명시
- 브라우저 보안 정책으로 인해 화질이 통일되지 않을 경우, 경고 문구 제공

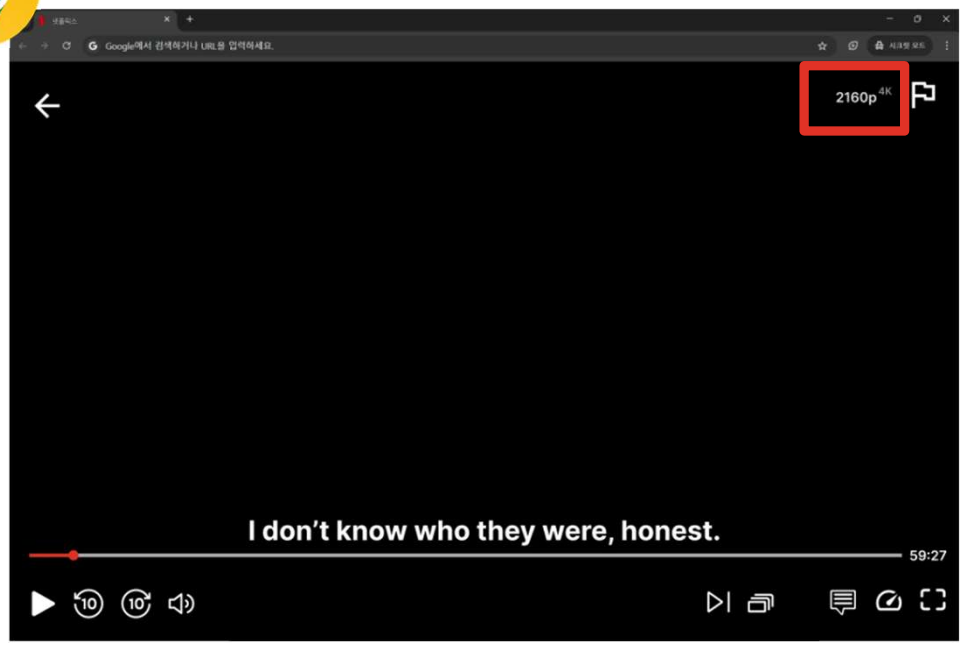
기능 2: 밝기 조절 개선

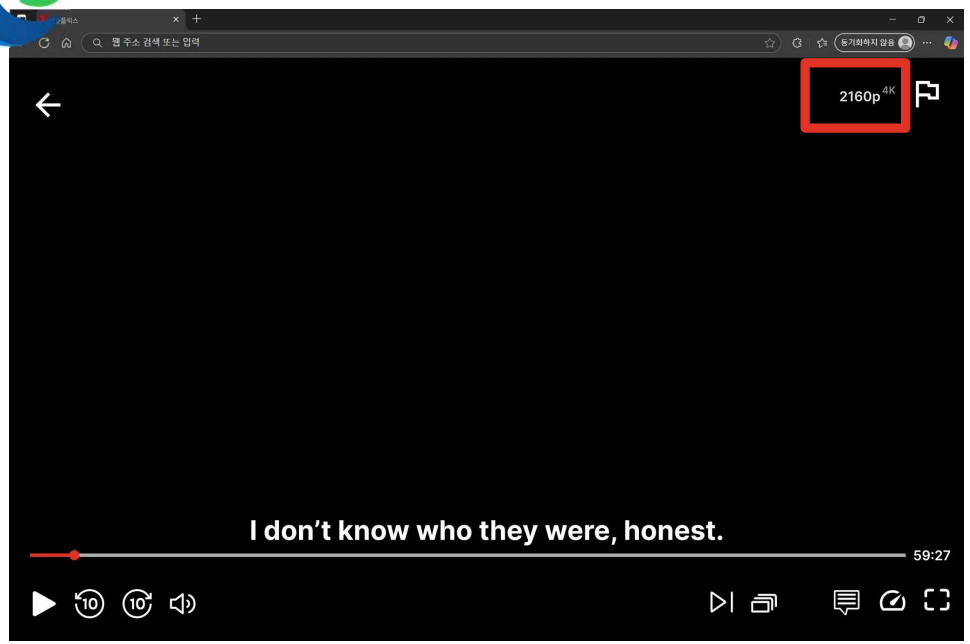
- 디바이스 기본 밝기 정보와 스트리밍 UI의 밝기 조절 기능을 연동
- 밝기 정책에 대한 안내 문구 제공

기능 3: 복수 자막 지원

- 두 개 이상의 언어를 동시에 시청 가능
- 다국적 사용자, 외국어 학습층의 요구 충족

04. 주요 화면 프로토타입

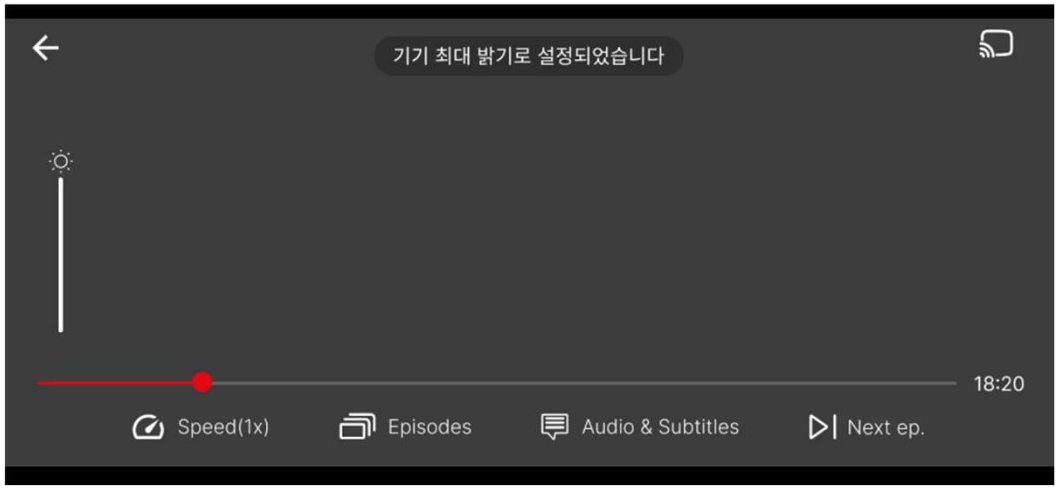


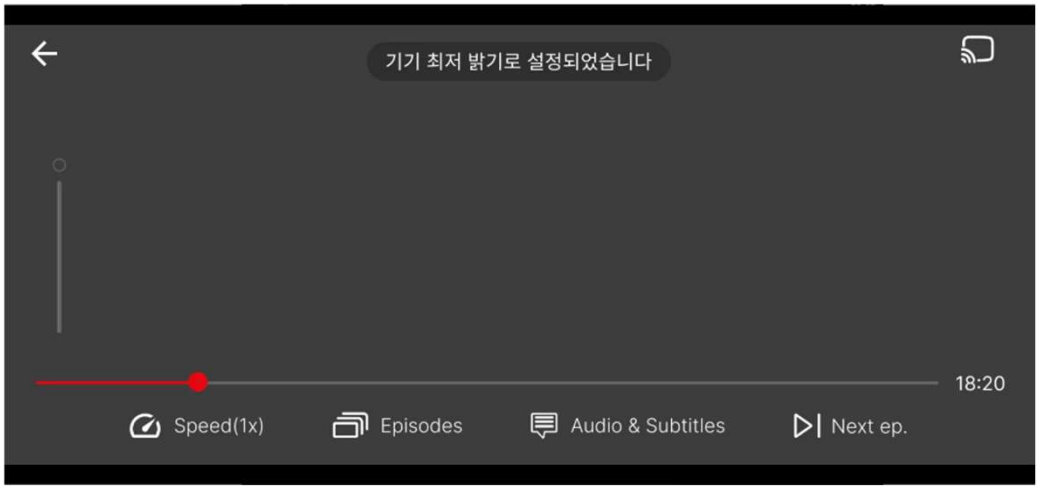


화질 명시

- 현재 사용자가 제공 받튼 컨텐츠의 화질 정보를 스트리밍 시스템 UI에 명시함
- 이를 통해 사용자가 균일한 화질의 컨텐츠를 제공 받고 있음을 확인할 수 있음

04. 주요 화면 프로토타입

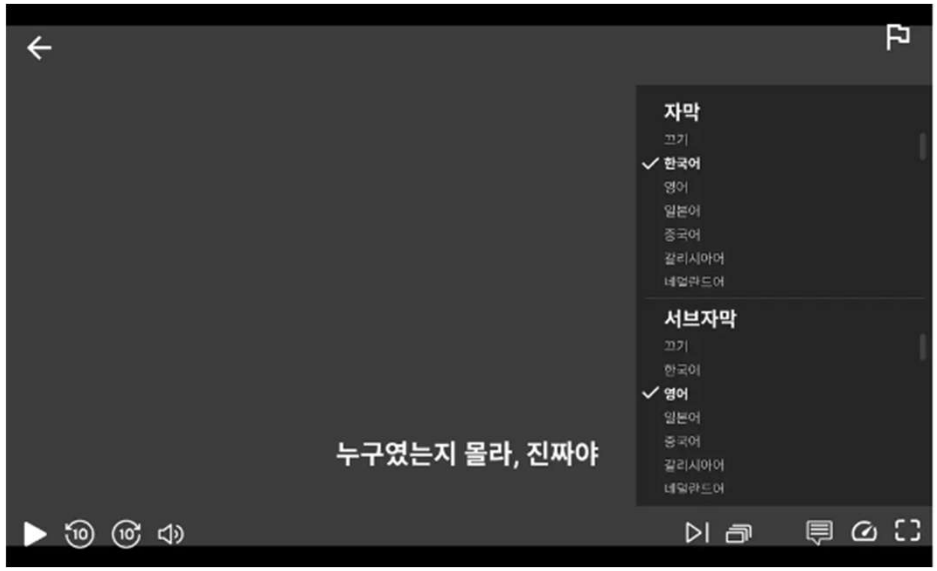


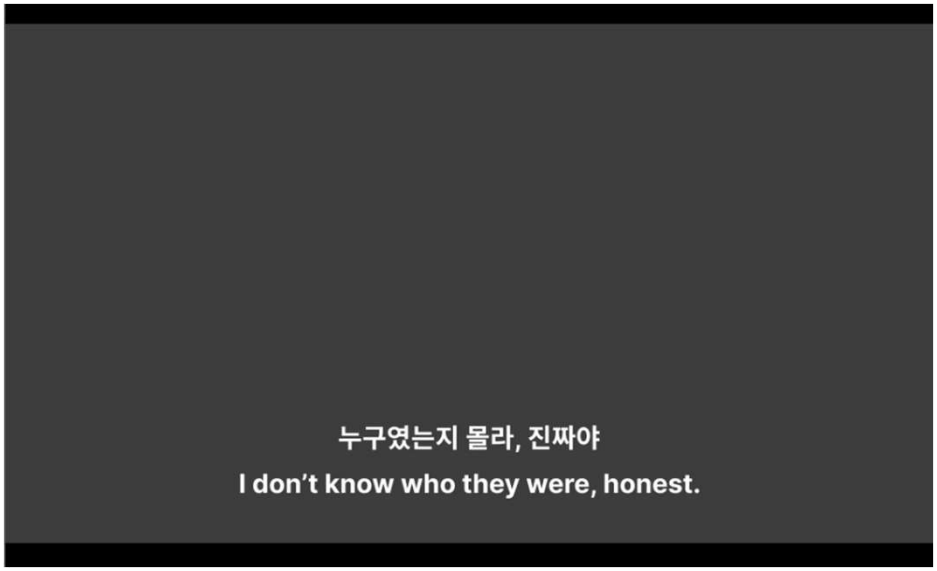


밝기 조절 문구

- 디바이스에 따라 연동된 밝기 조절 기능의 작동을 사용자에게 명시함

04. 주요 화면 프로토타입





복수 자막 지원

- 사용자는 복수 자막 언어를 선택할 수 있음
- 선택한 언어에 따라 자막이 2줄로 제시됨

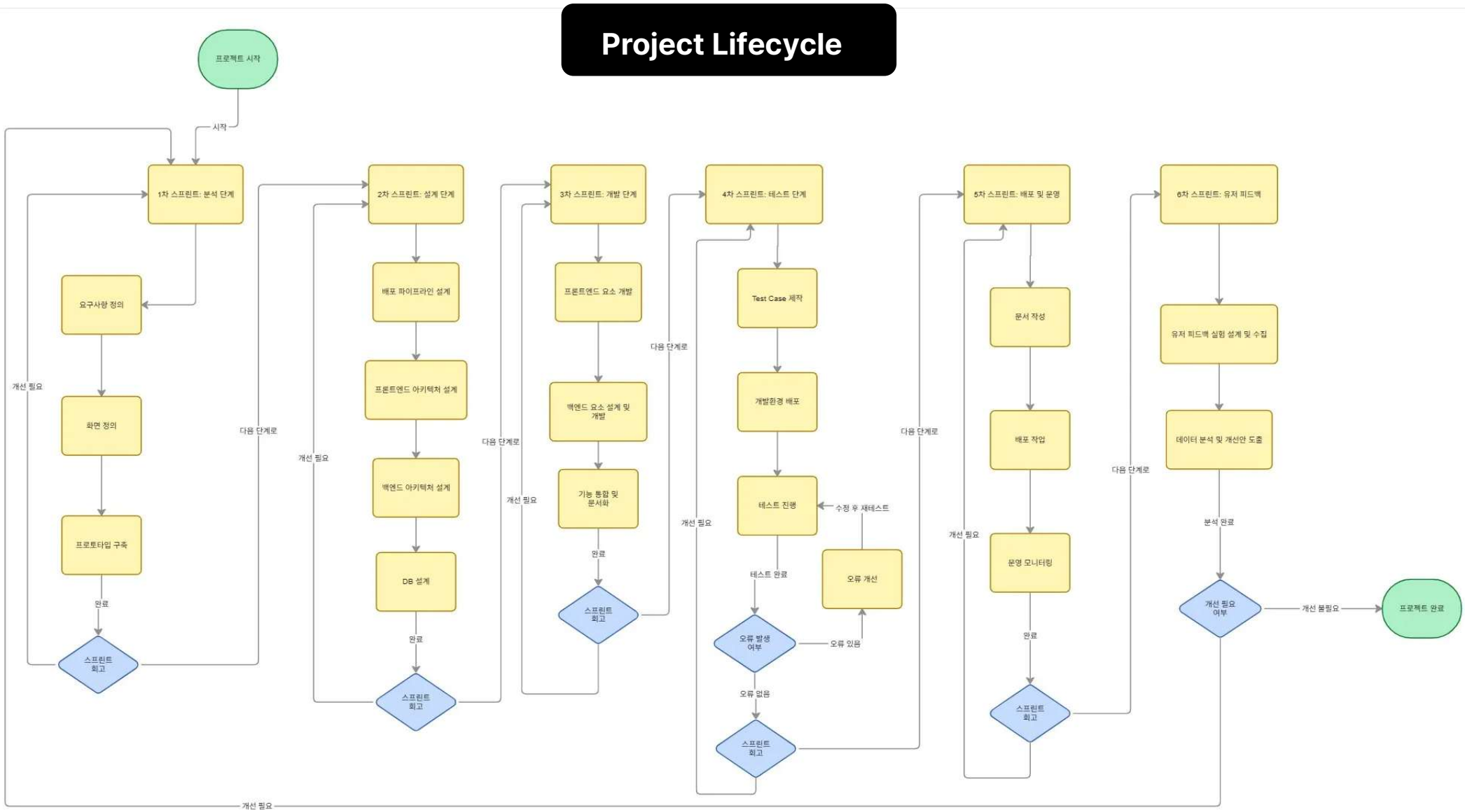
05. 추진 전략 및 체계 – 완성된 WBS

1.1	UI 개선 프로세스 1차	20%		26.01.01	26.01.31			이예준		
1.1.1	분석	4.0%		26.01.01	26.01.03				요구사항 정의서, 화면 정의서, 프로토타입, 프로토타입 검증 결과서	
1.1.1.1	요구사항 정의	2.0%		26.01.01 9:00	26.01.01 18:00					Jira (프로젝트 관리)
1.1.1.1.1	이해관계자 식별 및 인터뷰			26.01.01 9:00	26.01.01 12:00			이예준	인터뷰 결과 요약	Zoom (온라인 시)
1.1.1.1.2	기능/ 비기능 요구사항 분류 및 구체화			26.01.01 13:00	26.01.01 15:00			이예준	요구사항 목록(FR/NFR)	Jira
1.1.1.1.3	요구사항 우선순위 설정			26.01.01 15:00	26.01.01 16:00			김준성	우선순위 정리표	Jira
1.1.1.1.4	요구사항 정의서 작성 및 승인			26.01.01 16:00	26.01.01 18:00			김준성	승인된 요구사항 정의서	Confluence(팀 협업, 공유)
1.1.1.2	화면 정의	1.5%		26.01.02 9:00	26.01.02 18:00					
1.1.1.2.1	화면 흐름 및 네비게이션 설계			26.01.02 9:00	26.01.02 11:00			배형빈	화면 흐름도	Figma (협업 중심 디자인 도구)
1.1.1.2.2	와이어프레임 및 레이아웃 설계			26.01.02 11:00	26.01.02 12:00			배형빈	와이어프레임	Figma
1.1.1.2.3	UI/UX 시각 디자인			26.01.02 13:00	26.01.02 16:00			배형빈	UI 시안	Figma
1.1.1.2.4	화면 기능 및 규칙 정의			26.01.02 16:00	26.01.02 17:00			원대효	화면 기능 정의서	Confluence
1.1.1.2.5	화면 정의서 작성 및 승인			26.01.02 17:00	26.01.02 18:00			원대효	승인된 화면 정의서	Confluence
1.1.1.3	프로토타입 구축	0.5%		26.01.03 9:00	26.01.03 18:00					
1.1.1.3.1	시각화 도구 선정			26.01.03 9:00	26.01.03 10:00			이승영		
1.1.1.3.2	핵심 사용자 흐름 및 UI 구성			26.01.03 10:00	26.01.03 12:00			이승영	핵심 사용자 흐름 프로토타입	Figma
1.1.1.3.3	가상의 상호작용 구현			26.01.03 13:00	26.01.03 16:00			이한이	인터랙티브 프로토타입	Figma Prototype
1.1.1.3.4	요구사항 반영 여부 확인 및 승인			26.01.03 16:00	26.01.03 18:00			이한이	프로토타입 검증 결과	Jira, Confluence, Figma
1.1.2	설계	4.0%		26.01.04	26.01.10				UX flow/와이어프레임, UI 시안, 설계 리뷰 기록	
1.1.2.1	배포 파이프라인 설계	1.0%		26.01.04 09:00	26.01.05 11:00					
1.1.2.1.1	현재 배포 과정 분석			26.01.04 09:00	26.01.04 13:00			이예준	배포 프로세스 분석 문서, flow diagram	Confluence, Google docs, Jira
1.1.2.1.2	CI/CD 기준 수립			26.01.04 13:00	26.01.05 11:00			이예준	CD/CI 기준정의서, 배포 승인 정책 문서, 요구사항서	Notion, Jira, Jenkins UI
1.1.2.2	프론트엔드 아키텍쳐 설계	1.5%		26.01.05 13:00	26.01.07 13:00					
1.1.2.2.1	컴포넌트 구조 설계			26.01.05 13:00	26.01.06 11:00			김준성	FE 컴포넌트 구조도, Atomic design 구조 정의서	Figma, Confluence
1.1.2.2.2	상태관리 흐름 설계			26.01.06 11:00	26.01.06 18:00			김준성	상태관리 플로우 다이어그램	Whimsical, Draw.io, Figma
1.1.2.2.3	기술 스택 및 패턴 정의			26.01.07 9:00	26.01.07 13:00			배형빈	사용 라이브러리/도구 리스트, 개발 환경 구성 문서	Confluence, Notion
1.1.2.3	백엔드 아키텍쳐 설계	1.0%		26.01.07 13:00	26.01.09 16:00					
1.1.2.3.1	API 설계			26.01.07 13:00	26.01.08 14:00			배형빈	API 스펙 문서, 데이터 스키마	Swagger, JSON schema generator
1.1.2.3.2	기능 각본 설계			26.01.08 14:00	26.01.08 19:00			원대효	기능 Use Case 정의서, 시나리오 플로우 문서, 예외 케이스 핸들링 정의서	colfluence
1.1.2.3.3	ERD/시퀀스/데이터 플로우 설계 문서화			26.01.09 9:00	26.01.09 11:00			원대효	ERD, 데이터 정규화 문서, DFD	MySQL workbench, confluence, Draw.io
1.1.2.4	DB 설계	0.5%		26.01.09 11:00	26.01.10 18:00					
1.1.2.4.1	데이블 구조 설계			26.01.09 11:00	26.01.09 15:00			이승영	엔티티 목록 정의서, 데이터 접근 패턴 분석 문서	ERDCloud, confluence
1.1.2.4.2	인덱싱 전략 설계			26.01.09 15:00	26.01.10 11:00			이승영	인덱스 설계 문서, 쿼리 최적화 고려 사항 정리	confluence
1.1.2.4.3	데이터 흐름 정리			26.01.10 13:00	26.01.10 15:00			이한이	데이터 입력, 조회 흐름도, 수명주기 정리 문서	confluence
1.1.2.4.4	DB 스키마 문서화			26.01.10 15:00	26.01.10 18:00			이한이	최종 DB 스키마 정의서, ERD 수정본	confluence, google docs
1.1.3	개발	6.0%		26.01.11	26.01.20				프론트엔드 모듈/코드, 백엔드 모듈/코드, 코드 리뷰 결과 보고서	
1.1.3.1	프론트엔드 요소 개발	3.0%		26.01.11 09:00	26.01.13 18:00					
1.1.3.1.1	해상도 요소 컴포넌트 구현			26.01.11 09:00	26.01.12 18:00			이예준	UI 컴포넌트, 컴포넌트 코드	Figma, Jira
1.1.3.1.2	해상도 변경 이벤트 처리 로직 개발			26.01.13 09:00	26.01.13 18:00			김준성	이벤트 처리 코드	Jira, Confluence
1.1.3.2	백엔드 요소 설계 및 개발	2.0%		26.01.14 09:00	26.01.16 18:00					
1.1.3.2.1	해상도 관련 데이터 소스 정의			26.01.14 09:00	26.01.14 18:00			배형빈	ERD, 데이터 스키마	MySQL Workbench, ERDCloud
1.1.3.2.2	해상도 api 설계 및 구현			26.01.15 09:00	26.01.16 18:00			원대효	API 명세, API 코드	Swagger, Postman, Jira
1.1.3.3	기능 통합 및 문서화	1.0%		26.01.17 09:00	26.01.20 18:00					
1.1.3.3.1	프론트엔드-백엔드 요소 통합			26.01.17 09:00	26.01.18 18:00			이승영	연동된 기능, 통합 테스트 결과	Postman, Jira
1.1.3.3.2	작업내용 문서화			26.01.19 09:00	26.01.20 18:00			이한이	기술문서, 작업 보고 정리본	Confluence, Google Docs

05. 추진 전략 및 체계 – 완성된 WBS

1.1.4	**테스트**	4.0%		26.01.21	26.01.27				Test case sheet, 버그 리포트, QA 테스팅 결과 보고서	
1.1.4.1	Test Case 제작	1.0%		26.01.21 9:00	26.01.23 18:00					
1.1.4.1.1	테스트 범위 정의			26.01.21 09:00	26.01.21 18:00			이예준	테스트 범위 정의서, 테스트 대상 기능 목록	Jira(이슈·요구사항·작업을 관리하는 프로젝트 관리 도구)
1.1.4.1.2	정상 케이스(Test Case) 작성			26.01.22 09:00	26.01.22 18:00			김준성	정상 흐름 Test Case 문서	ChatGPT Enterprise (테스트 시나리오/Negative Case 초안 등을 만들어주는 AI 기반 테스트 설계 보조 도구)
1.1.4.1.3	예외 케이스(Negative Test Case) 작성			26.01.23 09:00	26.01.23 18:00			김준성	예외 상황 Test Case 문서	
1.1.4.2	개발환경 배포	0.5%		26.01.24 9:00	26.01.24 18:00					Spinnaker (배포 자동화 도구(CD))
1.1.4.2.1	개발 서버/로컬 환경 설정			26.01.24 09:00	26.01.24 13:00			배형빈	개발 환경 설정 문서, 개발용 설정 파일	Jenkins (CI(지속적 통합) 도구)
1.1.4.2.2	소스코드 빌드 및 배포			26.01.24 13:00	26.01.24 18:00			배형빈	빌드 결과물, 개발 서버 배포 완료 보고	Docker
1.1.4.3	테스트 진행	1.5%		26.01.25 09:00	26.01.26 18:00					
1.1.4.3.1	통합 테스트 실행			26.01.25 09:00	26.01.25 18:00			원대효	통합 테스트 결과 보고서	Cypress (웹 화면을 자동으로 테스트하는 프론트엔드 E2E 테스트)
1.1.4.3.2	시스템 테스트 실행			26.01.26 09:00	26.01.26 13:00			이승영	시스템 테스트 결과 보고서	Playwright (다양한 브라우저를 자동으로 조작하는 크로스 브라우저 테스트 도구)
1.1.4.3.3	인수 테스트 실행			26.01.26 13:00	26.01.26 18:00			이승영	인수 테스트 결과 및 승인 기록	Postman(API 요청을 보내고 응답을 검증하는 API 테스트·디버깅 도구)
1.1.4.4	오류 발생 시 개선	1.0%		26.01.27 09:00	26.01.27 18:00					
1.1.4.4.1	로그 분석 및 원인 파악			26.01.27 09:00	26.01.27 13:00			이한이	오류 로그 분석 결과, 원인 분석 보고서	
1.1.4.4.2	코드 수정 및 보완 작업 수행			26.01.27 13:00	26.01.27 18:00			이한이	수정된 코드, 수정 이력 문서	
1.1.5	**배포 및 운영**	1.0%		26.01.28	26.01.29				배포 로그 및 버전 릴리즈 노트, 운영 서버 반영 결과 확인서, hotfix 내역	
1.1.5.1	문서 작성	0.5%		26.01.28 09:00	26.01.29 12:00					
1.1.5.1.1	사용자 가이드 작성			26.01.28 09:00	26.01.28 11:00			이예준	사용자 가이드 문서	Confluence, Google Docs
1.1.5.1.2	운영 정책 문서 작성			26.01.28 11:00	26.01.28 15:00			이예준	운영 정책 문서	Confluence
1.1.5.1.3	관리자 문서 작성			26.01.28 15:00	26.01.28 18:00			김준성	관리자용 기능 설명 문서	Confluence
1.1.5.1.4	cs 및 장애 대응 가이드 작성			26.01.29 09:00	26.01.29 10:30			배형빈	CS 대응 가이드, 장애 대응 절차서	Confluence
1.1.5.1.5	시나리오 완료 보고서			26.01.29 10:30	26.01.29 12:00			배형빈	기능 시나리오 실행 결과 보고서	Confluence, Google Docs
1.1.5.2	배포 작업	0.3%		26.01.29 13:00	26.01.29 15:30					
1.1.5.2.1	서버 설정			26.01.29 13:00	26.01.29 14:30			원대효	서버 설정값, 환경 설정 문서	Jenkins, Docker
1.1.5.2.2	서비스 배포			26.01.29 14:30	26.01.29 15:30			원대효	배포 완료 결과	Spinnaker, Jenkins, Docker
1.1.5.3	운영 모니터링	0.2%		26.01.29 15:30	26.01.29 18:00					
1.1.5.3.1	모니터링 도구 연동			26.01.29 15:30	26.01.29 17:00			이승영	모니터링 도구 연동 설정	모니터링 시스템, Jira
1.1.5.3.2	서버 모니터링 대시보드 확인			26.01.29 17:00	26.01.29 18:00			이한이	서버 상태 확인 결과	모니터링 도구 대시보드
1.1.6	**유저 피드백 확인**	1.0%		26.01.30	26.01.31				피드백 수집 요약 리포트, 개선사항 우선순위 목록	
1.1.6.1	유저 피드백 실험 설계 및 수집	0.6%		26.01.30 9:00	26.01.31 11:00					
1.1.6.1.1	실험 가설 및 비교군 설정			26.01.30 9:00	26.01.30 10:30			이예준	A/B 테스트 가설 문서, 대조군·비교군 정의	Confluence, Jira
1.1.6.1.2	사용자 그룹 분리 및 화면 노출 설정			26.01.30 10:30	26.01.30 15:30			김준성	그룹 분리 기준, 노출 설정 결과	Jira, Confluence
1.1.6.1.3	사용자 행동 데이터 수집			26.01.30 15:30	26.01.31 11:00			배형빈	사용자 행동 로그 데이터	모니터링/Analytics 도구, Jira
1.1.6.2	데이터 분석 및 개선안 도출	0.4%		26.01.31 11:00	26.01.31 18:00					
1.1.6.2.1	데이터 시각화 및 분석			26.01.31 11:00	26.01.31 15:00			원대효	시각화된 그래프/차트, 분석 결과	Confluence, Google Docs
1.1.6.2.2	개선사항 정리 및 제안			26.01.31 15:00	26.01.31 18:00			이승영	개선안 리스트, 제안 문서	Confluence, Google Docs

06. 추진 전략 및 체계 - 프로젝트 일정



06. 추진 전략 및 체계 - 프로젝트 일정

1) 기획 및 요구분석

- **요구사항 정의 :** 이해관계자 식별 및 인터뷰 진행. 화질 명시 및 통일성 강화, 밝기 조절 문구, 복수 자막 기능에 대한 기능/비기능 요구사항 확정. 요구사항 정의서 작성.
- **화면 정의** : 화질 표시 아이콘, 밝기 조절 슬라이더, 이중 자막 레이아웃 UI/UX 및 와이어프레임 설계
- **프로토타입 구축** : 핵심 사용자 프로토타입 시각화

2) 설계

- **배포 파이프라인 설계 :** 현재 배포 과정을 분석하고 CI/CD 기준을 수립
- **프론트엔드 아키텍쳐 설계** : 브라우저별 해상도 감지 및 상태관리 화면 정의
- **백엔드,DB 설계** : 밝기 및 자막 설정을 위한 API 설계. DB 스키마 문서화

3) 개발

- **프론트엔드 요소 개발** : 실제 화질 확인 및 밝기 연동, 자막 이중 렌더링 컴포넌트 구현
- **백엔드 개발** : 화면, 밝기, 자막의 데이터 소스 정의 및 API 구현
- **기능 통합** : UI와 백엔드 데이터 연동 및 통합

4) 테스트

- **Test case 제작** : 브라우저별 화질을 확인할 수 있는지, 밝기 및 자막이 정상 작동하는지 TC 작성
- **테스트 진행** : 다양한 디바이스 및 브라우저 환경에서 통합 테스트 수행 및 오류 수정
- **오류 발생 시 개선 :** 로그 분석 및 원인 파악, 코드 수정 및 보완 작업 수행

5) 배포 및 운영

- **문서 작성** : 사용자 가이드 작성(FAQ 등), 운영 정책 문서, 관리자 문서, cs 및 장애 대응 가이드 작성
- **배포 작업** : 서버 설정 및 실제 서비스에 신규 UI 배포, 기존 시스템과 충돌 여부 확인하면서 안정화
- **운영 모니터링** : 배포 후 화질 경고 문구 노출 빈도 및 시스템 부하 모니터링 대시보드 점검.

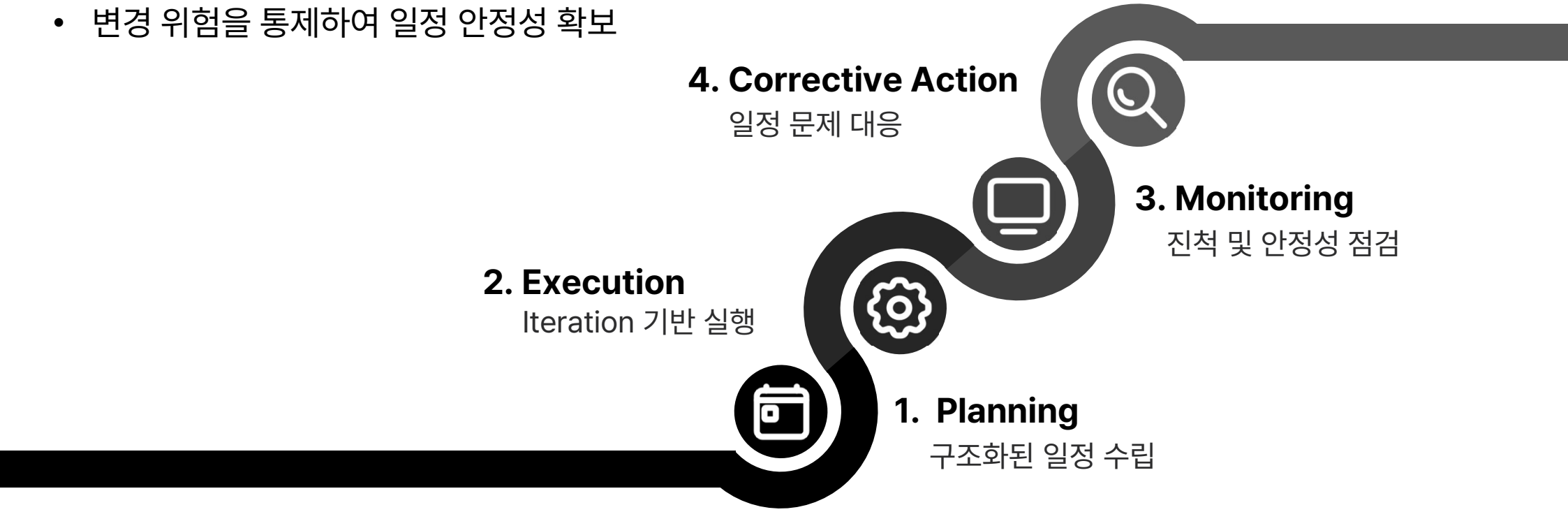
6) 유저 피드백 확인

- **실험 설계 및 수집** : 신규 기능 사용자 그룹 분리 및 행동 데이터 수집
- **데이터 분석** : 복수 자막 사용의 편리성, 밝기 조절 연동의 정확성 등에 대한 사용자 반응 데이터를 분석.
- **개선안 도출 :** 기능 사용률 분석 및 데이터 시각화를 통해 UI 가시성 및 편의성 개선안 도출. 개선안 정리 및 향후 프로세스 제안

07. 추진 전략 및 체계 – 일정 관리 방안 개요

목적 (Purpose)

- 예측 가능한 프로젝트 운영
- 변경 위험을 통제하여 일정 안정성 확보



07. 추진 전략 및 체계 – 일정 관리 방안 상세

1. Planning

① WBS 및 활동 분해

- WBS는 월 단위 UI 개선 프로세스(1~12차) 기준
- 각 차수는 동일 반복 구조
  분석 → 설계 → 개발 → 테스트 → 배포 → 피드백

② 마일스톤 정의

- 마일스톤 : 월별 UI 개선 프로세스 완료(1~12차)
- 완료 기준 : 분석/설계/개발/테스트/배포 산출물
  + 유저 피드백 확보

③ 활동 순서

- 고정된 선행 관계 사용
  분석 → 설계 → 개발 → 테스트 → 배포 → 피드백 → 다음 차수 분석

④ 기간 산정

- 1차 실적 확보 후 → 이후 차수는 Velocity 기반 조정
- 필요 시 PERT로 기간 타당성 검증

⑤ 기준선(Baseline)

- 모든 차수는 1개월 Timebox
- 변경은 승인 절차를 통해서만 반영

2. Execution

① Iteration 운영

- 모든 차수는 Sprint처럼 반복 운영
- 1차는 상세 계획, 이후는 Rolling Wave로 점진적 상세화

② 책임자 지정

- 모든 WBS 태스크에 담당자 매핑
- 담당자 공백 시 상위 책임자 자동 커밋

③ 주간 점검

- 진행률 업데이트
- 위험, 병목 공유
- 다음 주 계획 조정

07. 추진 전략 및 체계 – 일정 관리 방안 상세

3. Monitoring

① 실적 추적

- 일정(%), 리소스, 산출물 품질을 WBS 기반으로 지속 추적

② 진척률 관리

- 진행률은 WBS 가중치 기반
- 일정 안정성 확인
  1. SPI(EV/PV)
  2. Planned vs Actual (%)

③ 마일스톤 리뷰

- 차수 종료시 완료 여부 확인
- 결함 및 피드백 반영 상태 점검
- 통과 시 다음 차수 시작

④ 일정 편차 대응

- 지연 발생 시 Critical Path 확인
- 필요 시 시정조치 적용

4. Corrective Action

- 계획대로 복귀하기 위한 조치 실행

1. Fast-tracking : 병렬 진행
2. Crashing : 인력 추가 투입
3. Scope 조정 : Timebox 유지 위해 기능 축소
4. Risk 대응 : 이슈 전환 후 즉시 처리

데이터 관리

- 산출물 및 일정 변경 기록은 Jira/ Confluence 저장
- 진척 데이터는 차수별 분석에 활용

메트릭

- 핵심 지표 : SPI, Planned vs Actual, Velocity, 결함 추세
- 용도 : 일정 성과 파악 및 다음 차수 예측

변경관리 연동

- 요구사항, 디자인 변경 발생시 일정 영향 분석
- 필요 시 기준선(Baseline) 갱신

08. 추진 전략 및 체계 – 변경관리 절차

시나리오 1. 고객의 변경 요청

a. 고객 제보 단계 *고객센터 웹페이지 / 모바일 앱*

- 고객이 제품 사용 중 오류나 불편사항을 발견. -> 제보사항을 올림.

b. 고객센터 처리 단계 *CRM 시스템 (Dynamics 365 등)*

- 시스템이 자동으로 결함 유형을 분류 -> 담당자 검토 후 보고서 생성 -> PM에게 전달

c. 변경 여부 검토 단계 *내부 이슈 관리 시스템 (Notion, Teams 등)*

- PM은 변경의 필요성, 효용, 비용을 평가

d. 변경 계획 수립 단계 *GitLab/Jira/TestRail/CI-CD 환경 등*

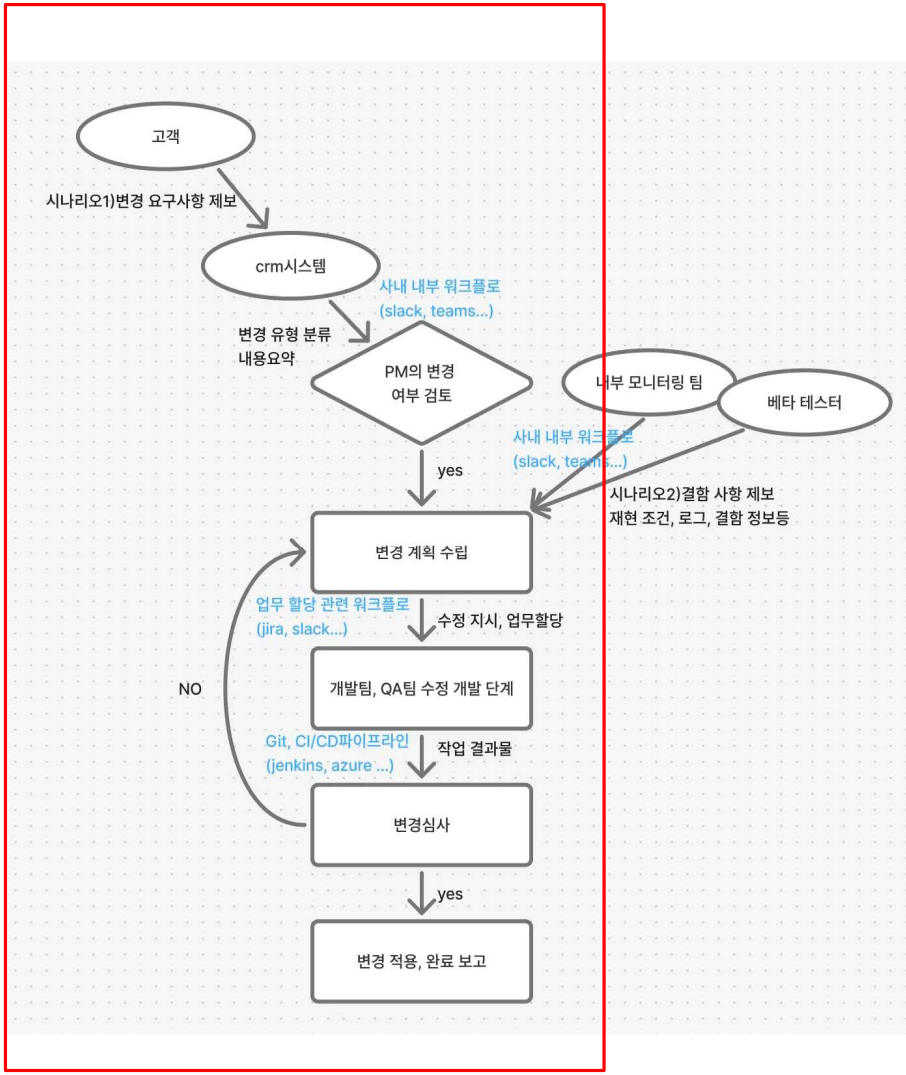
- PM은 결함의 심각도, 영향 범위, 긴급도 평가
- 산하 개발 QA팀에 수정 요청

e. 개발 · QA 수정 단계 *ServiceNow, Jira Service등 변경관리모듈*

- QA팀이 테스트 기준 정의
- 수정 및 테스트 진행
- 결과를 PM에게 보고

e. 적용 및 완료 보고 *Jenkins, GitHub Actions, Spinnaker 등*

- PM이 수정 내역 검토 -> 컨펌 후 운영 서버에 배포



08. 추진 전략 및 체계 – 변경관리 절차

시나리오 2. 결함 발견

a-1. 모니터링 팀 결함 제보 단계 *ELK 스택, Atlas 등의 모니터링 툴*

- 모니터링 팀이 결함 세부 정보(로그, 메트릭, 재현 조건)를 기록.
- 제보 내용이 Jira나 CRM 시스템으로 자동 접수되며, Slack 채널로 알림 전송.

a-2. 사내 베타 테스터의 결함 제보 *내부 이슈 관리 시스템 (Notion, Teams 등)*

- 베타 버전 테스트 중 발생한 이슈에 대해 결함 원인, 변경 사유, 변경 범위 등을 명시하여 변경 요청서(RFC)를 작성

b. 변경 계획 수립 단계 *GitLab/Jira/TestRail/CI-CD 환경 등*

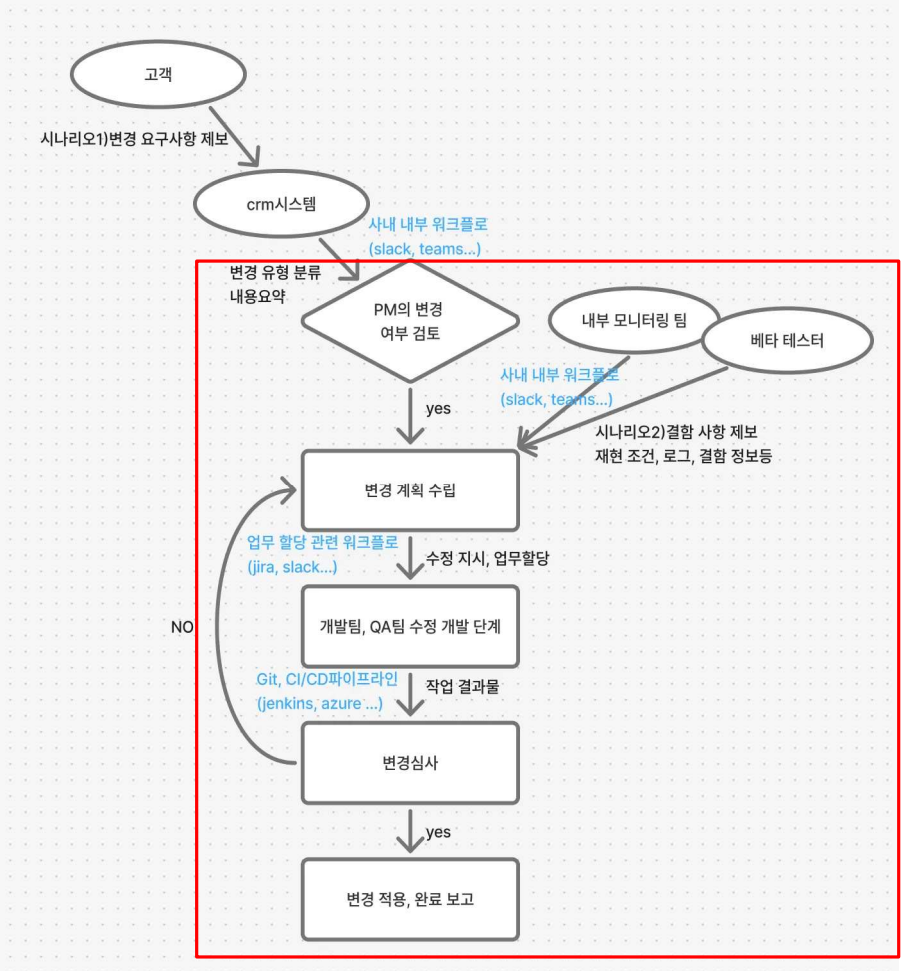
- PM은 결함의 심각도, 영향 범위, 긴급도 평가
- 산하 개발 QA팀에 수정 요청

c. 개발 · QA 수정 단계 *ServiceNow, Jira Service등 변경관리모듈*

- 서비스 운영 시간을 고려하여 변경 시행 일정 수립
- 변경에 필요한 인력과 스케줄 조율 및 이해관계자에게 사전 통지
- 변경 작업에 대한 절차와 시나리오 문서화 (예, 롤백 계획)

d. 적용 및 완료 보고 *Jenkins, GitHub Actions, Spinnaker 등*

- 엔지니어가 변경 사항을 반영한 코드를 배포
- -> 위험도에 따라 자동 반영 혹은 관리자의 검토 요청



09. 추진 전략 및 체계 – 품질 관리 방안

1. 동료 검토

- 필요성 및 목적
  - 해상도, UI/UX 품질, API 연동 등은 프론트 엔드와 백 엔드 모두의 품질에 영향을 미침.
  - 단일 팀 검토로는 한계가 있기 때문에 여러 팀이 함께 상호 검증해 통합 단계의 품질과 일관성을 확보함.
- 수행 시기
  - 프론트 엔드 – 백 엔드 통합 단계
- 대상
  - 프론트 엔드 코드, 백 엔드 코드, 디자인 결과물
- 방법
  - 동료 검토 (같은 프로젝트에 참여하는 동료들이 서로의 결과물을 검토해 품질을 향상시키는 과정)
- 주체
  - 각 분야 개발자, PM
- 도구
  - Click UP, Jira

2. 제품 메트릭 수집

- 필요성 및 목적
  - 제품 품질 개선을 데이터 기반으로 검증하기 위해 사용자 상호작용 중심의 정량 메트릭 수집 체계 도입.
  - 기능 개선이 실제 사용자 경험 향상으로 이어졌는지를 데이터로 검증하고, 향후 개선 방향을 근거 있게 제시.
- 수행 시기
  - 운영 모니터링 단계
- 대상
  - 배포 완료된 프로토 타입 제품
- 방법
  - 유저 기반 메트릭 수집 (사용자의 실제 행동 데이터를 상호작용 분석해 제품 풀질을 개선하는 과정)
- 주체
  - 모니터링팀
- 도구
  - Grafana Labs , Google Analytics

3. 정적 코드 분석

- 필요성 및 목적
  - 개발된 코드를 실행하기 전에 자동으로 검사해 보안, 버그, 비효율 코드 패턴을 사전에 탐지.
  - 이를 통해 코드 품질 지속적으로 관리, 안정적이고 신뢰성 높은 소프트웨어 개발 지원.
- 수행 시기
  - FE·BE 개발 단계 동시·지속 수행
- 대상
  - 프론트 엔드 코드, 백 엔드 코드
- 방법
  - 정적 분석 도구를 사용해 코드를 실행하지 않고 소스 레벨에서 검사(보안, 효율성, 결함)
- 주체
  - 전체 개발팀
- 도구
  - SonarQube, Cursor

09. 추진 전략 및 체계 – 동료검토

ClickUp

- 개요

- 팀의 업무와 프로젝트를 한곳에서 체계적으로 관리할 수 있는 통합 협업 플랫폼..

- 주요 기능

- 업무 관리: 작업 생성, 담당자 지정 등을 통해 팀 전체의 업무 흐름을 시각적으로 관리.
- 리뷰·피드백 관리: 다양한 팀이 동일한 공간에서 리뷰 일정을 공유하고 피드백을 남길 수 있기 때문에 부서 간 협업 효율을 향상.

Jira

- 개요

- 개발 프로젝트의 이슈, 코드 리뷰, QA 결과 등을 통합적으로 관리할 수 있는 협업 및 프로젝트 관리 플랫폼.

- 주요 기능

- **이슈 및 작업 관리**: 버그, 기능 추가, 코드 리뷰 등 다양한 작업을 티켓 단위로 등록하고 담당자 및 진행 단계를 명확히 관리.
- **리뷰·피드백 추적**: 코드 리뷰, QA, 디자인 검토 등의 과정을 상태별로 관리하며, 리뷰 기록과 피드백을 일원화하여 품질 향상에 기여.

09. 추진 전략 및 체계 – 제품 메트릭 수집

Grafana Labs

- 개요

- 시스템 로그, 지표, 사용자 데이터를 실시간으로 시각화하여 모니터링할 수 있는 분석 도구.

- 주요 기능

- **실시간 모니터링**: 서버 상태, API 응답 시간, 사용자 트래픽 등의 지표를 대시보드 형태로 시각적으로 제공.
- **데이터 시각화**: 다양한 그래프·차트를 통해 이상 징후를 빠르게 파악하고 성능 저하 요인을 분석.

Google Analyitics

- 개요

- 웹·앱 사용자 행동을 자동으로 수집하고 분석하여 서비스 개선에 활용할 수 있는 사용자 행동 분석 플랫폼.

- 주요 기능

- **행동 자동 분석**: 클릭, 페이지 이동, 체류시간 등 주요 사용자 이벤트를 자동 수집 및 시각화.
- **전환 분석**: 기능 개선이 실제 사용자 경험 향상으로 이어졌는지를 데이터 기반으로 검증.

09. 추진 전략 및 체계 – 정적 코드 분석

sonarqube

- 개요

- 개발된 코드를 자동으로 분석해 **버그, 보안 취약점, 비효율적인 코드 패턴**을 탐지하는 정적 분석 도구.

- 주요 기능

- **코드 품질 분석**: 코드 내 결함, 보안 취약점, 중복 코드, 복잡도 등을 자동으로 검사하여 품질 지표 제공.
- **지속적 품질 관리**: CI/CD 파이프라인과 연동되어 코드 변경 시마다 자동 검사 수행, 품질 저하를 실시간으로 방지.

cursor

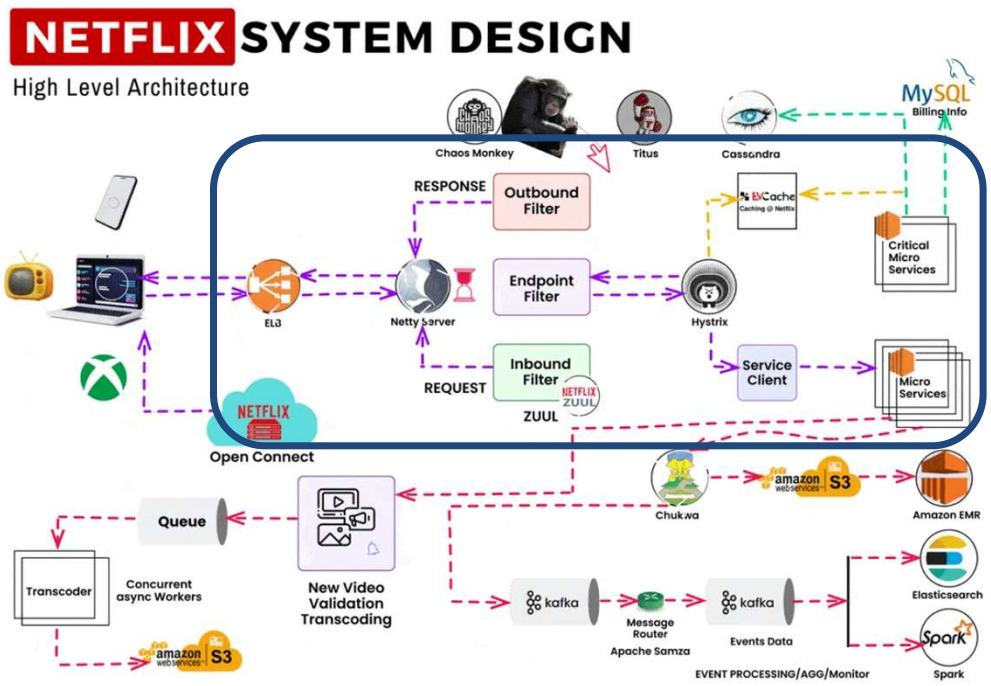
- 개요

- 웹AI 기반 코드 분석 및 리뷰 보조 도구로, 개발자가 작성한 코드를 **자동으로 이해·리팩터링·리뷰**할 수 있도록 지원하는 도구.

- 주요 기능

- **AI 코드 리뷰**: 코드 구조와 로직을 분석하여 오류 가능성, 비효율적인 구현 패턴 등을 자동 피드백.
- **리팩터링 지원**: 코드 가독성과 유지보수성을 높이기 위한 개선안을 자동으로 제안.

10. 추진 전략 및 체계 – 인프라 구축 방안: 기존 넷플릭스 아키텍쳐



요청 정보 전송

- ELA

- 전 세계 사용자 요청을 적절한 서버로 전송

- Outbound Filter

- 응답을 돌려줄 때까지 추가 가공 수행

- Endpoint Filter

- 요청이 라우팅 된 후 어떤 서비스로 보낼지 판단

- Hystrix

- 특정 서비스에서 장애 발생 시 회로 차단
  => 전체 서비스로 번지는 것 방지

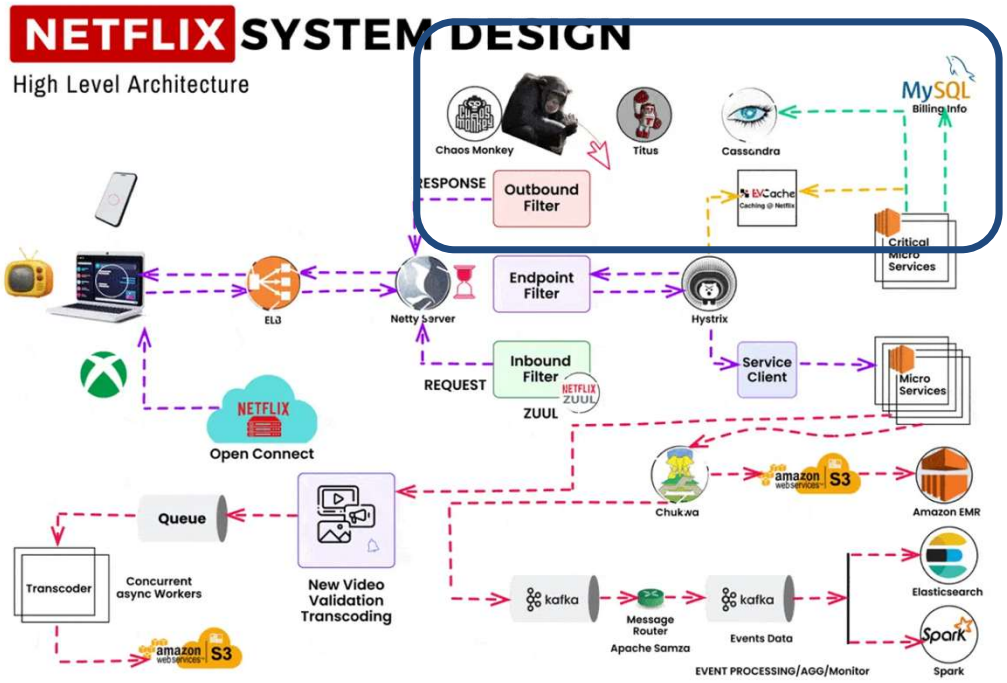
- Micro Services

- 외부 요청이 마이크로 서비스로 분산됨 (추천 서비스, 사용자 계정 서비스 등)
- -> AWS에서 백엔드를 실행하여 오토 스케일링 관리

- zuul

- 넷플릭스의 API GateWay로 인바운드 필터를 먼저 통과
- -> 인증 검증, 요청 형태 검증 등 MSA로 보내기 전 필터링 단계

10. 추진 전략 및 체계 – 인프라 구축 방안: 기존 넷플릭스 아키텍쳐



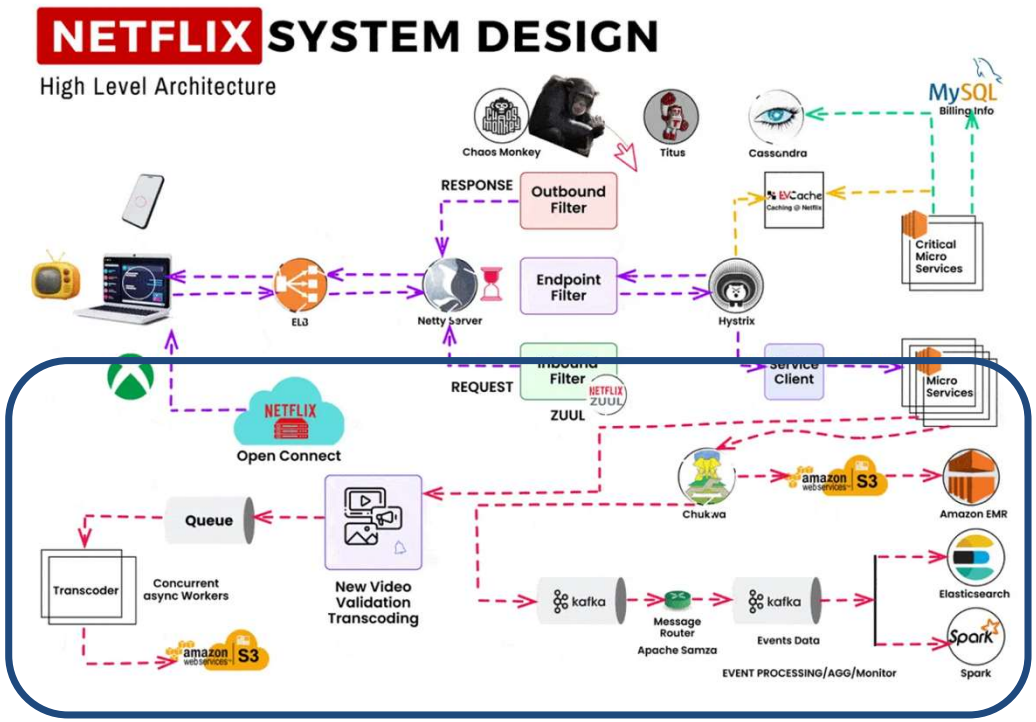
부하 테스트

- Chaos Monkey 부하 테스트 도구
  - 일부 서버를 의도적으로 죽여 시스템이 잘 살아남는지, 복원력을 테스트하는 넷플릭스 내부 도구
- Titus 컨테이너 오케스트레이션 플랫폼
  - 마이크로 서비스 컨테이너를 배포하고 자동 스케일링 및 모니터링 담당

DB

- Cassandra (NoSQL) -> 사용자 프로필 / 시청 기록 / 플레이백 상태 등 기록
- MySQL -> 결제 같은 민감하고 강한 정합성이 필요한 데이터 기록
- EVCache(메모리 캐싱) -> 자체 캐싱 계층

10. 추진 전략 및 체계 – 인프라 구축 방안: 기존 넷플릭스 아키텍쳐



이 과정에서 생성되는 메타 데이터 확보 필요

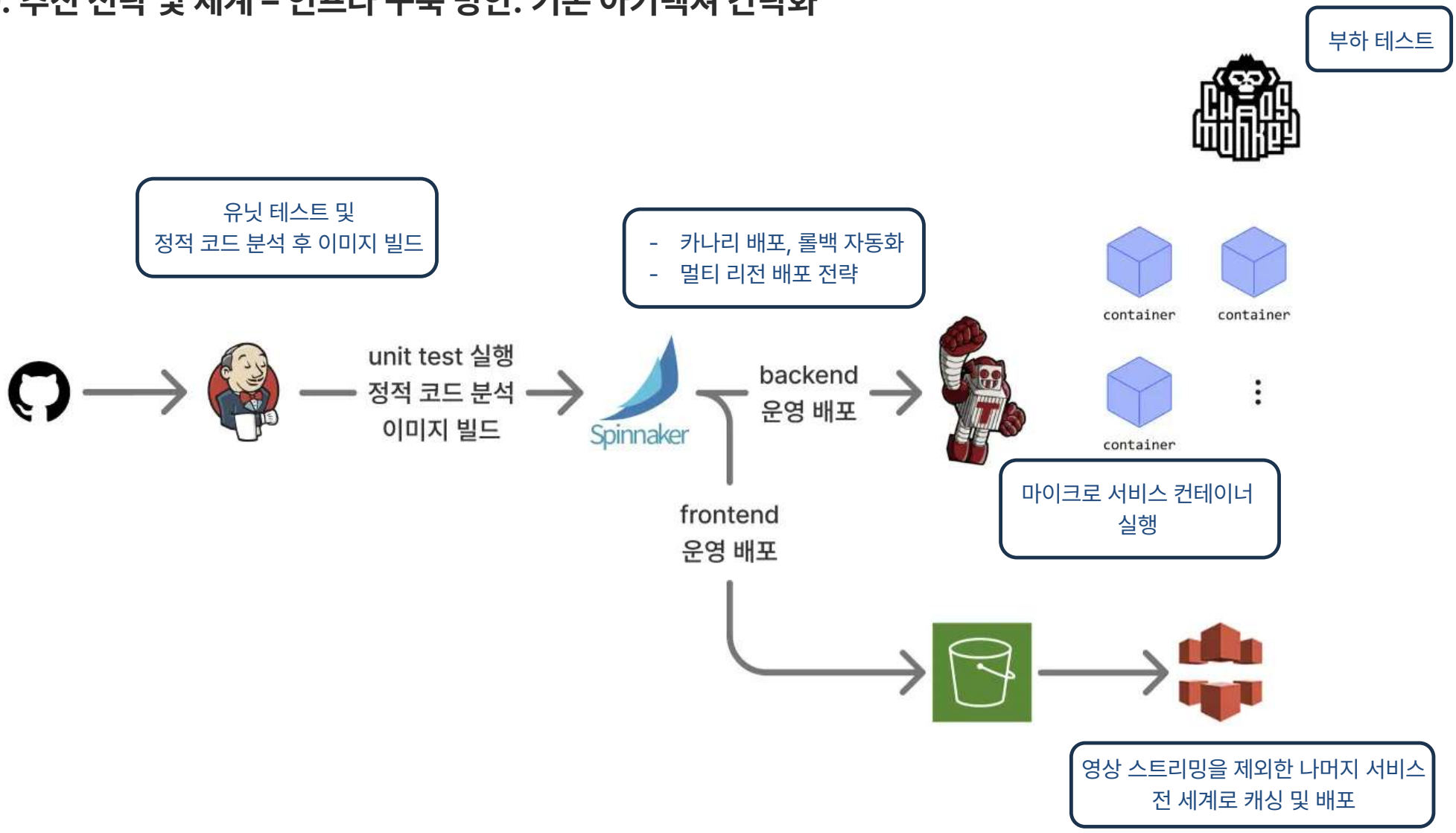
영상 정보 인코딩

- transcoder -> 영상을 여러 화질로 변환 (1080, 720, 480, 모바일 최적화)
- Amazon S3 -> 변환된 영상 저장
- open connect appliance -> 자체 구축한 CDN으로 전송비용절감 및 품질 극대화

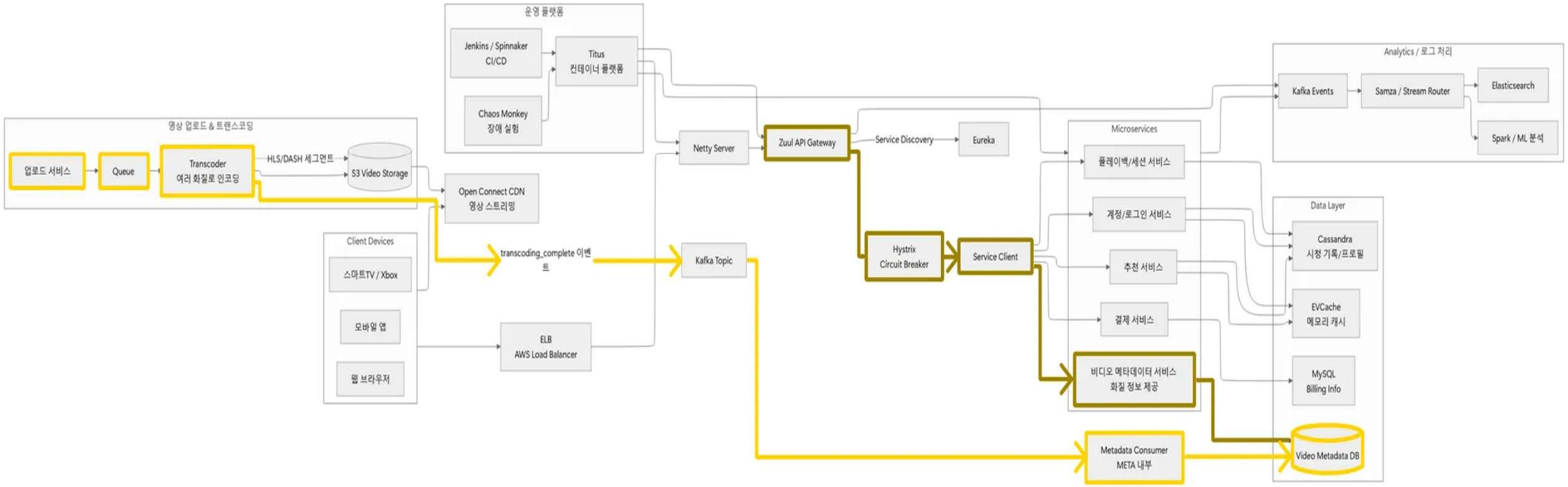
사용자 로그 저장

- chukwa, amazon, s3, emr -> 로그 저장 및 대규모 분석 처리
- Kafka 기반 로그 처리 -> 모든 데이터가 kafka 이벤트 스트림으로 모임
- samza(stream processing) -> 실시간으로 데이터 라우팅 및 가공
- eleasticsearch -> 검색 및 로그 모니터링에 사용
- spark -> 대규모 머신러닝 / 데이터 분석

10. 추진 전략 및 체계 – 인프라 구축 방안: 기존 아키텍쳐 간략화



10. 추진 전략 및 체계 – 인프라 구축 방안: 해상도 기능 구현 과정 반영



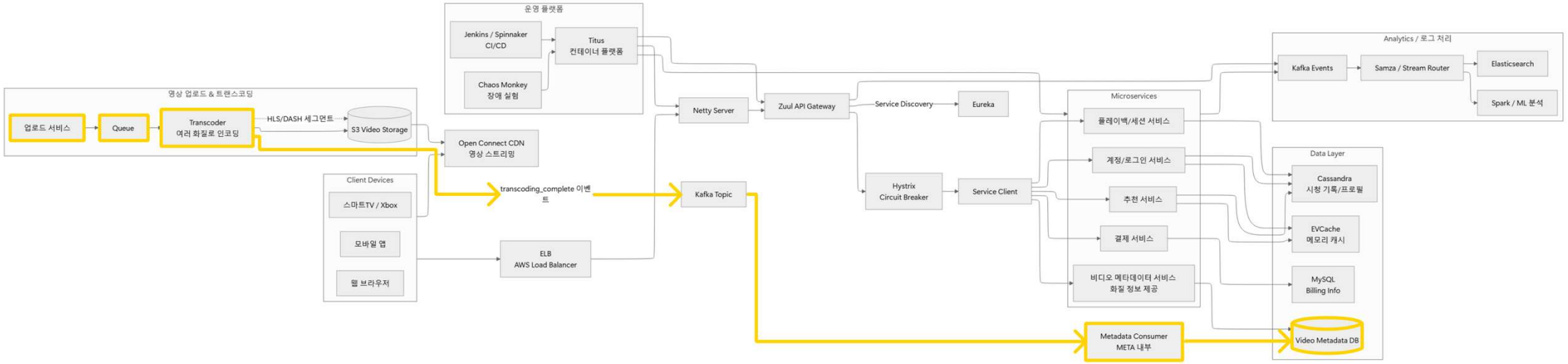
DB에 영상에 대한 메타 정보 (해상도 포함)를 저장하는 과정

프론트에서 요청 시 앞서 저장한 DB에서 메타정보를 가지고 와 응답하는 과정

10. 추진 전략 및 체계 – 인프라 구축 방안: 해상도 기능 구현 과정 반영

메타 데이터 저장 과정

업로드 서비스 → Queue → Transencoder → Kafka Topic → Video meat DB

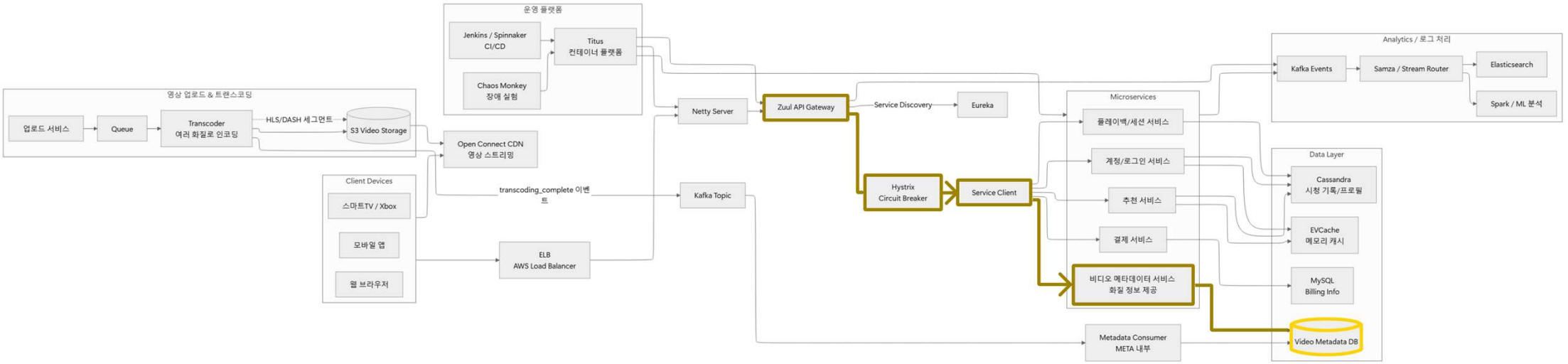


DB에 영상에 대한 메타 정보 (해상도 포함)를
저장하는 과정

10. 추진 전략 및 체계 – 인프라 구축 방안: 해상도 기능 구현 과정 반영

디바이스에 응답 과정

Zuul API Gateway → Hystrix Circuit Breaker → Service Client → 비디오 메타데이터 관리 서비스 → Video Metadata DB

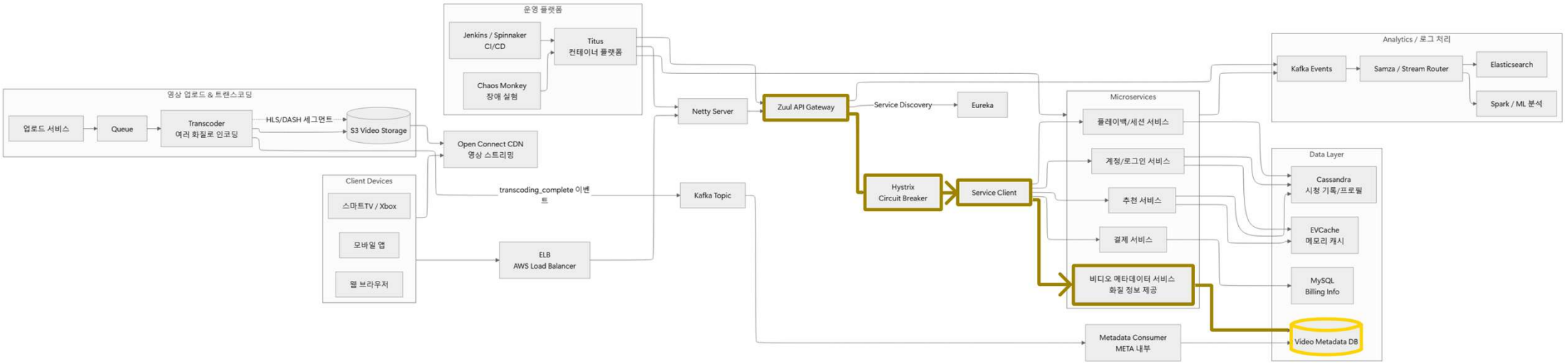


프론트에서 요청 시 앞서 저장한 DB에서 메타정보를 가지고 와 응답하는 과정

10. 추진 전략 및 체계 – 인프라 구축 방안: 해상도 기능 구현 과정 반영

디바이스에 응답 과정

Zuul API Gateway → Hystrix Circuit Breaker → Service Client → 비디오 메타데이터 관리 서비스 → Video Metadata DB



프론트에서 요청 시 앞서 저장한 DB에서 메타정보를 가지고 와 응답하는 과정

11. 팀원 소개 및 프로젝트 소감

팀원명: 김준성

직책: 팀장

장점

1. 아마추어 PT 가능
2. 팀 리더 직책 수행 가능
3. 중저음 목소리 보유
4. 타인의 고충을 잘 들어줌
5. 거의 대부분의 음식을 잘 먹음

Liked – 좋았던 점, 마음에 들었던 점

강의 수강 전 25년 상반기에 진행했던 인턴에서 활동으로 체험했던 것들을 이론으로 배울 수 있어 개념을 재정립 할 수 있는 유익한 시간이었음

Learned – 배운 것

WBS 작성법, 프로젝트 일정 관리, Agile 기법 적용 이 가장 인상적이고 기억에 남는 내용이었음

Lacked – 부족했던 점, 아쉬웠던 점

기획 중심의 수업이다보니, 실제 프로그램 제작까지 연장되지 못했던 부분이 아쉬웠음

Longed for – 바랐던 점, 있었으면 좋았을 것

Agile 등의 프로젝트 플래닝 기법 중에서 실제 현업에서 적용되고 있는 사례를 더 많이 접하면 좋을 것 같음

11. 팀원 소개 및 프로젝트 소감

팀원명: 이예준

직책: 팀원

장점

1. 자료 정리를 잘하는 편임

2.일정은 최대한 맞추려고 노력함

3.타인의 이야기를 잘 듣고 배려하려고 함

4. 새로운 내용을 금방 이해하고 자연스럽게 받아들이는 편임

5. 맡은 일은 끝까지 성실하게 해내려 함

Liked – 좋았던 점, 마음에 들었던 점

여러 개발 프로세스를 한눈에 비교해 전체 흐름을 쉽게 이해할 수 있었고, SQA의 중요성과 프로세스 품질이 제품 품질로 이어지는 개념이 명확했으며, 구글 CI/CD나 DORA 지표 같은 실무 사례 덕분에 현업 감을 얻을 수 있었고, 메트릭 종류와 활용법이 구체적으로 제시되어 품질 판단 기준을 이해하기 쉬웠다.

Learned – 배운 것

품질은 프로세스에서 나온다는 기본 원칙을 확실히 이해하게 되었고, QA와 QC의 차이와 역할을 명확히 구분할 수 있었으며, 다양한 메트릭이 실제 관리 도구로 활용된다는 점과 DevOps에서 리드타임·배포빈도 같은 지표가 중요한 이유를 알게 되었다.

Lacked – 부족했던 점, 아쉬웠던 점

팀 프로젝트를 통해 요구사항부터 품질 검토까지의 전체 흐름을 경험하긴 했지만, 유즈케이스에서 테스트 케이스로 자연스럽게 이어지는 실전 예시가 조금 더 있었으면 좋았겠다.

Longed for – 바랐던 점, 있었으면 좋았을 것

실제 데이터를 활용해 메트릭을 분석하고 품질 지표를 직접 해석해볼 수 있는 실습이 추가된다면 전체적인 이해가 더 깊어질 것 같다.

11. 팀원 소개 및 프로젝트 소감

팀원명: 이승영

직책: 팀원

장점

1. 주어진 일은 끝까지 최선을 다함

2.타인의 의견을 잘 받아들임

3. 의견이 있을 땐 적극적으로 얘기함

4. 구조화를 잘함

5. 배워나가는 것을 좋아함

Liked – 좋았던 점, 마음에 들었던 점

프로젝트를 할 때 항상 급하게 진행하는 경우가 많았는데 프로젝트의 관리 방법에 어떤 것들이 있는지 알고 실제 팀 프로젝트에 적용해볼 수 있어 좋았다. 개발 직전까지의 과정만 설계하다보니 부담없이 프로젝트 관리에 초점을 맞추어, 수업에서 배운 내용을 적용해볼 수 있어 좋았다.

Learned – 배운 것

Agile은 그냥 유명한, 트렌디한 협업 방식인 줄 알았는데 이 방식이 왜 사용되기 시작했고, 어느 상황에 적합하고 그렇지 않은지 자세히 알 수 있어 좋았다. Agile 외에도 다양한, 전문적인 프로젝트 관리 기법이 있다는 게 신기했다. 실제 기업에서도 복잡한 코드 구조 때문에 많이 고민을 하고 있다는 점을 깨달았고 코드 뿐만 아니라 전체 프로젝트의 관리를 깔끔하게 해야 되겠다는 다짐을 하게 됐다.

Lacked – 부족했던 점, 아쉬웠던 점

기존 서비스를 발전시키는 방식으로 과제가 진행되다 보니, 처음부터 기획을 체계적으로 쌓아가는 경험을 충분히 하지 못한 점이 아쉬웠다

Longed for – 바랐던 점, 있었으면 좋았을 것

보통 동아리에서 프로젝트를 할 때는 빨리빨리 기능을 구현해야 하는 경우가 많아서, 수업에서는 그동안 해보지 못했던 기획 및 팀 관리에 초점을 두고 과제를 진행할 수 있다면 좋을 것 같다. 또한 프로젝트 관리 도구를 팀플에 직접 적용해볼 수 있는 기회도 마련된다면 좋을 것 같다.

11. 팀원 소개 및 프로젝트 소감

팀원명: 원대호

직책: 팀원

장점

1. 매사에 긍정적이고 적극적임
2. 상대방의 장점과 특성을 잘 파악함
3. 계획 작성을 바탕으로 시간 관리를 잘함
4. 잠을 엄청 잘 잠
5. 자료 조사를 잘 함

Liked – 좋았던 점, 마음에 들었던 점

팀장을 중심으로 팀원 간 협업이 원활하게 이루어졌고, 소프트웨어 개발 과정을 단계별로 학습할 수 있어 좋았습니다.

Learned – 배운 것

요구사항 분석부터 설계, 테스트까지의 전체 개발 프로세스를 보다 심도 있게 이해할 수 있었습니다.

Lacked – 부족했던 점, 아쉬웠던 점

실제 서비스로 확장했을 때의 실현 가능성에 대해 충분히 논의하지 못한 점이 아쉬웠습니다.

Longed for – 바랐던 점, 있었으면 좋았을 것

도출한 Pain Point를 실제 코드로 구현해보는 실습 시간이 추가됐다면 이해가 더욱 깊어졌을 것 같습니다.

11. 팀원 소개 및 프로젝트 소감

팀원명: 이한이

직책: 팀원

장점

1. 남의 말을 잘 들어줌

2. 다이어그램 제작 가능

3. 꼼꼼함

4. 긍정적인 편

5.새로운 것에 도전하는 것을 좋아함

Liked – 좋았던 점, 마음에 들었던 점

팀원들과 협업하는 과정이 즐거웠습니다. 팀원들과 소통하면서 완성도있게 프로젝트를 마무리할 수 있었던 것 같습니다.

Learned – 배운 것

체계적인 소프트웨어 개발 프로세스의 전체적인 흐름을 이해할 수 있었고, 요구사항 정의서나 WBS 등의 문서를 작성하는 구체적인 방법을 익힐 수 있었습니다.

Lacked – 부족했던 점, 아쉬웠던 점

수업 시간 내에 배운 내용을 모두 적용해 보기에는 실습 시간이 다소 부족하여, 과제를 급하게 마무리해야 했던 점이 아쉬웠습니다.

Longed for – 바랐던 점, 있었으면 좋았을 것

이론적인 설명뿐만 아니라 실제 현업에서 사용되는 문서 양식이나 프로젝트 성공/실패 사례 등 더 구체적인 실무 예시를 접하고 싶습니다.

11. 팀원 소개 및 프로젝트 소감

팀원명: 배형빈

직책: 팀원

장점

1. 남의 의견을 잘 이해할 수 있음
2. 톡톡 튀는 생각을 잘함
3. 의견을 비유를 들어 잘 설명함
4. 널널함
5. 사람을 긍정적으로 봄

Liked – 좋았던 점, 마음에 들었던 점

팀 프로젝트 팀원들이 모두 적극성이 높고, 팀플 경험이 많은지 커뮤니케이션 능력이 뛰어났습니다.
수업 내용이 흥미로웠습니다. 컴퓨터 자체에 대한 내용만큼이나 중요한 제품 개발 프로세스에 대한 이해를 할 수 있었습니다.

Learned – 배운 것

회의할 때 커뮤니케이션 능력을 가장 많이 배웠습니다. 인원이 많은데도 모두가 의견을 편하게 낼 수 있도록 조율을 잘 해가며 회의하는 모습이 인상적이었습니다.

Lacked – 부족했던 점, 아쉬웠던 점

시험 때 종이에 강의내용을 적어올 수 있게 하셨는데, 그냥 강의자료 대부분을 작게 만들어서 프린트해오는 경우가 많았던 것 같습니다. 손글씨만 허용하든, 아예 오픈북을 하는게 좋을 것 같습니다.

Longed for – 바랐던 점, 있었으면 좋았을 것

현재 기업들의 개발프로세스나 프로그램 사용 예시.

End of
Document