

구름톤 트레이닝 풀스택 1회차

# 온기종기 스터디 Ver 2.0

Computer Science

Problem Solving

Problem Based Learning

OGJC

오지고 지리는 생기발랄한 온기종기 스터디

# OGJG CONTENT

intro. 스터디 과정 살펴보기

- 제 1장 팀 소개
- 제 2장 Problem Solving
- 제 3장 Computer Science
- 제 4장 Problem Based Learing
- 제 5장 스터디 회고



### Problem Solving

- ✓ 팀원들의 풀이 설명이 한번에 이해가 안돼서 설명을 재요청했는데 불편해하지 않고 이해시키려고 노력해줘서 정말 감사했다.

✓ 스터디 시작 전에, 내 상황을 수치화 할 만한 지표가 없어서 전보다 얼마나 성장했는지 알 수 없어서 아쉽다.

✓ 알고리즘 문제 선정 시 같은 난이도의 문제를 선정했었어만 풀이 속도의 발전 과정을 알 수 있다는 사실을 회고하면서 깨닫게 되어 아쉽다.



### Computer Science

- ✓ github - wearesoft 에 나와있는 커리큘럼대로 진행했는데 모든 팀원들이 그것만 학습해온게 아니라 각자 추가적으로 조사한 자료를 공유해줘서 좋았다.

✓ 리뷰 날마다 호기심이 생겼던 부분들에 대해서 과할 정도로 의문을 많이 제기했는데 팀원들이 귀찮아하지 않고 팀원들도 의욕에 불타 같이 찾아주고 해결해줘서 감사했다.

✓ 원래 계획에 있던 데이터베이스와 디자인 패턴을 공부하지 못하고 스터디를 끝낸게 아쉽다.



# 제 1장. 팀 소개

팀장

김준서

팀원

한승재  
조재균  
안병규  
이정준  
이동진

주제

- Problem Solving
- Computer Science
- Problem Base Learning

## Problem Solving

백준 기준 실버 1~2 난이도 문제를 30분 안에 풀 수 있도록 알고리즘 문제 풀이 역량 향상

## Computer Science

기술 면접을 대비해 CS 기초 지식을 습득하고 공부한 지식을 모아 각자 CS 문서를 생성

## Problem Base Learning

개인 PBL 과제를 통해 역량을 쌓고,  
팀 PBL 과제를 하면서 기업 연계 프로젝트에 대비하기 위한  
Git 전략, 코딩 컨벤션 등의 협업 규칙들을 정하여 숙달 및 문서화

# 제 2장. Problem Solving

풀이 시간을 최대 2시간으로 제한

풀이 시간 기록

1

매주 같은 종류의 알고리즘 문제 3개 선정  
쉬움(실버3~5), 보통(실버1~2), 어려움(골드3~5)

2

해당 주간에 스터디에서 정해진 문제를 푼다.

3

그 다음 주 월요일,  
사다리를 통해 선정된 발표자는 본인의 풀이과정을 설명한다.

4

청취자들은 발표자의 코드를 피드백한다.  
만약 본인과 다른 풀이과정이라면 설명한다.

# 제 2장. Problem Solving - 13개의 알고리즘



Stack

Binary Search

Implement

Two Pointer

Prefix Sum

DFS

BFS

Backtracking

Dijkstra

Bellman-Ford

Kruskal - MST

Prim - MST

# 제 2장. Problem Solving - Dijkstra

## 7월 24일 알고리즘 리뷰 - Dijkstra

### ✓ [백준] 1753번: 최소 비용 구하기 - 골드 5

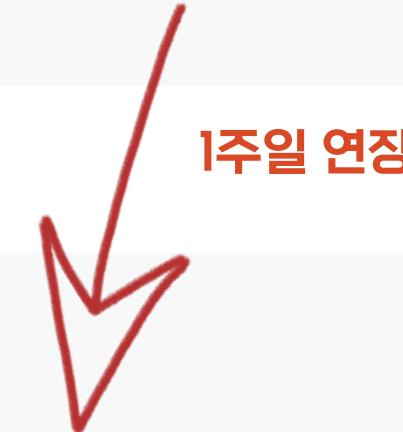
- 김준서 : 33분
- 한승재 : 80분
- 조재균 : 40분
- 안병규 : 35분
- 이정준: 50분
- 이동진 : 시간 초과

## 7월 31일 알고리즘 리뷰 - Dijkstra

### ✓ [백준] 1753번: 최단 경로 - 골드 4

- 김준서 : 9분
- 한승재 : 30분
- 조재균 : 42분
- 안병규 : 15분
- 이정준: 25분
- 이동진 : 시간 초과

1주일 연장



1차 시도



## 오늘의 리뷰

문제	[백준] 1916번 : 최소비용 구하기	[백준] 1446번: 지름길	[백준] 18352번: 특정 거리의 도시 찾기
난이도	🥇 골드 5	🥈 실버 1	🥈 실버 2
풀이시간 / 이름	33분 / 김준서	57분 / 김준서	시간 초과 / 김준서
풀이시간 / 이름	40분/ 조재균	시간초과/ 조재균	50분 / 조재균
풀이시간 / 이름	80분 / 한승재	시간초과 / 한승재	27분 / 한승재
풀이시간 / 이름	35분 / 안병규	90분 / 안병규	40분 / 안병규
풀이시간 / 이름	50분 / 이정준	시간초과 / 이정준	35분 / 이정준
풀이시간 / 이름	시간초과 / 이동진	시간초과 / 이동진	70분 / 이동진
선정자	조재균	안병규	이정준
발표자	이동진	안병규	한승재

## 고민 사항

- 다익스트라 시간복잡도  $E \log V$ 와  $V \log V$  어떤게 맞는 것인가? ( $E$  - 간선의 개수,  $V$  - 노드의 개수)
  - 간선의 개수가 많으면  $E \log V$ 에 가까워지고 적으면  $V \log V$ 에 가까워진다

### 다익스트라 알고리즘의 시간 복잡도

$V$ : 노드의 숫자,  $E$ : 간선의 숫자이다.

기본 알고리즘의 시간복잡도는  $O(V^2)$ 이다.

# 제 2장. Problem Solving - 백준 플래티넘 도전

## ♣ 오늘의 리뷰

문제	[백준] 1219번: 오민식의 고민	[백준] 1865번: 월홀	[백준] 11657번: 타임머신
난이도	🥇 플래티넘 5	🥇 골드 3	🥇 골드 4
풀이시간 / 이름	110분 / 김준서	시간초과 / 김준서	시간초과 / 김준서
풀이시간 / 이름	시간초과 / 조재균	42분 / 조재균	시간초과 / 조재균
풀이시간 / 이름	시간초과 / 한승재	45분 / 한승재	30분 / 한승재
풀이시간 / 이름	시간초과 / 안병규	시간초과 / 안병규	69분 / 안병규
풀이시간 / 이름	시간초과 / 이정준	110분 / 이정준	시간초과 / 이정준
풀이시간 / 이름	시간초과 / 이동진	시간초과 / 이동진	시간초과 / 이동진
선정자	전체	전체	전체
발표자	이동진	김준서	조재균

회고하고 다시보니 풀만한데?

# 제 2장. Problem Solving - 1차 측정

문제 수 : 2



골드 3~5



- 김준서 : 72분      풀 문제 수 2
- 한승재 : 85분      풀 문제 수 2
- 조재균 : 75분      풀 문제 수 1
- 안병규 : 40분      풀 문제 수 1
- 이정준 : 35분      풀 문제 수 1
- 이동진 : 시간초과      풀 문제 수 0

문제 수 : 4



실버 1~2



- 김준서 : 50분      풀 문제 수 2
- 한승재 : 40분      풀 문제 수 2
- 조재균 : 45분      풀 문제 수 2
- 안병규 : 45분      풀 문제 수 2
- 이정준 : 45분      풀 문제 수 2
- 이동진 : 70분      풀 문제 수 1

문제 수 : 2



실버 3~5

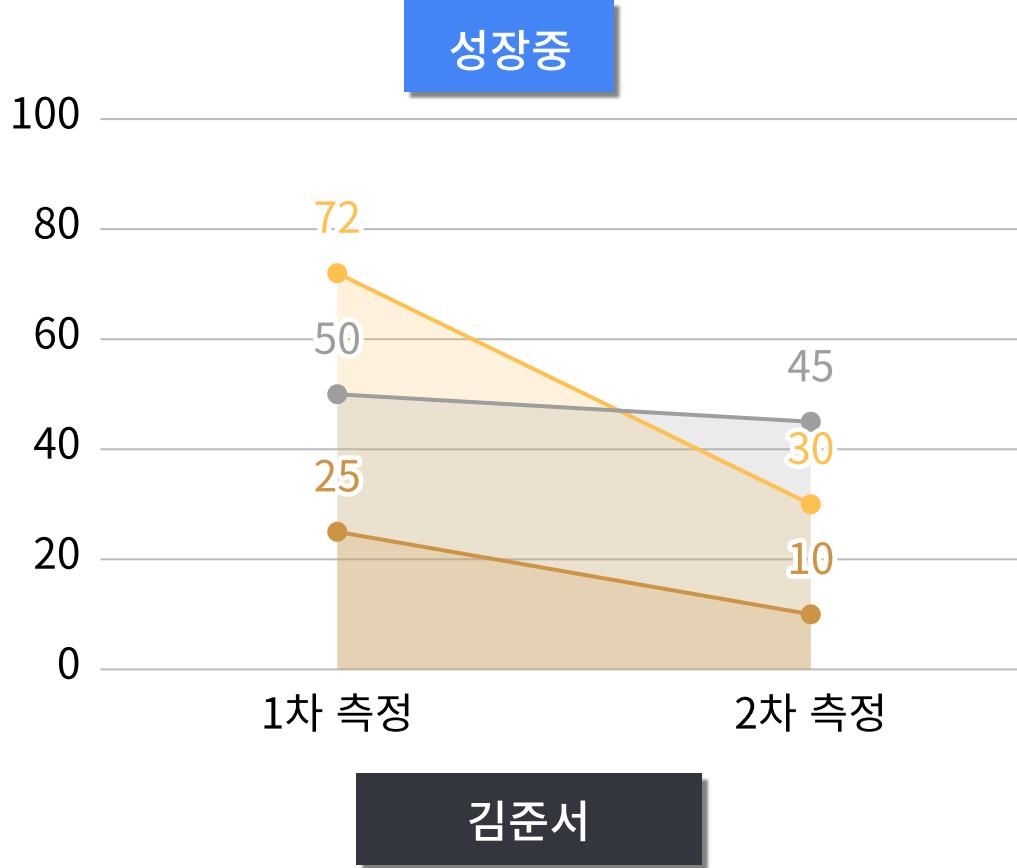


- 김준서 : 40분      풀 문제 수 2
- 한승재 : 21분      풀 문제 수 2
- 조재균 : 35분      풀 문제 수 2
- 안병규 : 37분      풀 문제 수 2
- 이정준 : 25분      풀 문제 수 2
- 이동진 : 60분      풀 문제 수 2

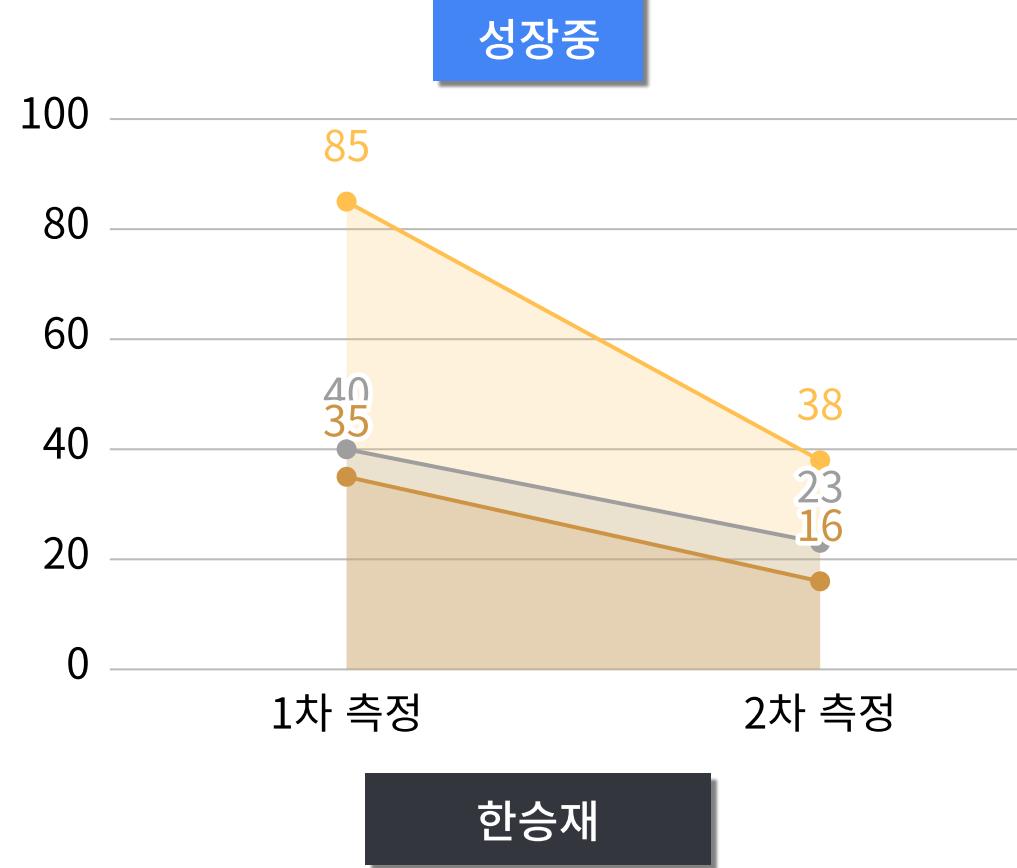
# 제 2장. Problem Solving - 2차 측정



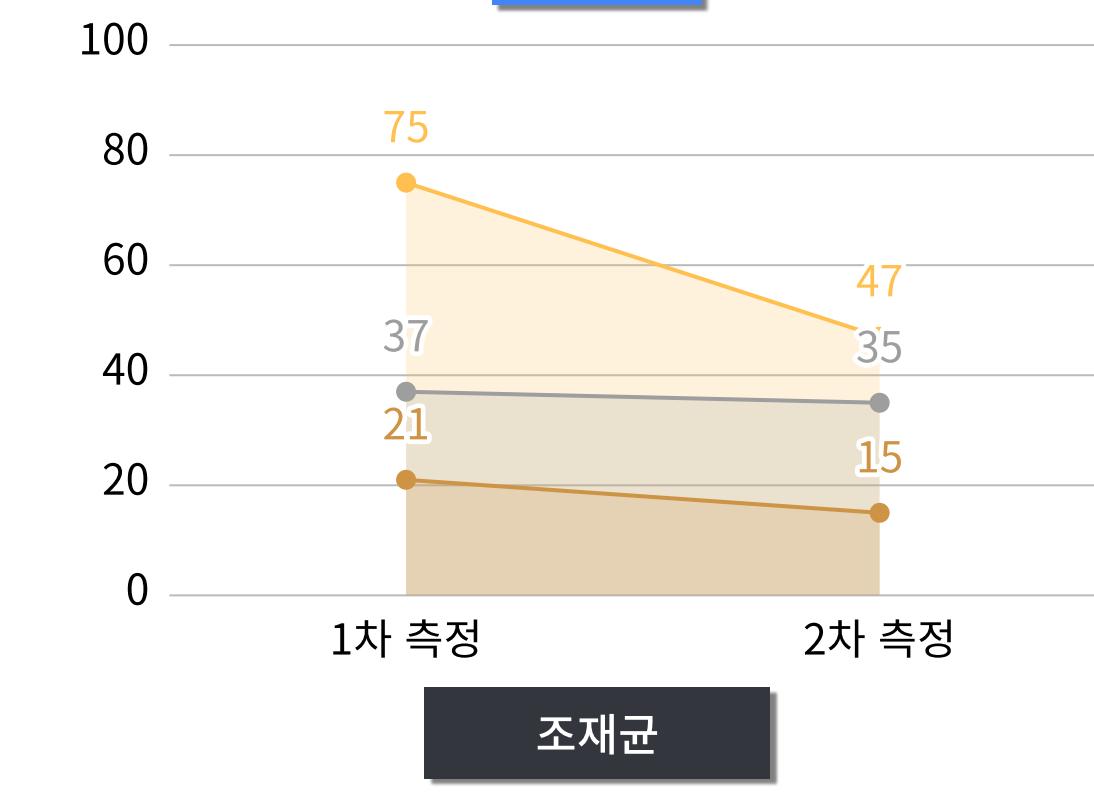
● 골드 3~5 ● 실버 1~2 ● 실버 3~5



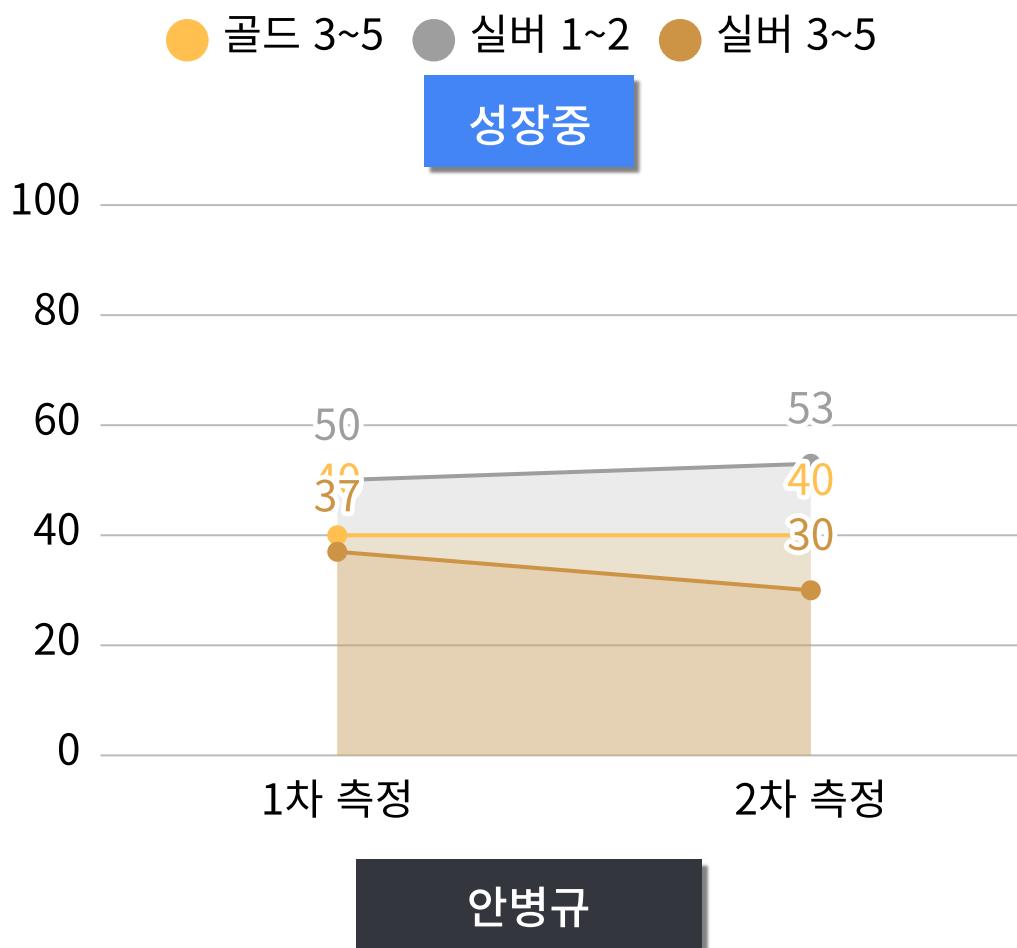
● 골드 3~5 ● 실버 1~2 ● 실버 3~5



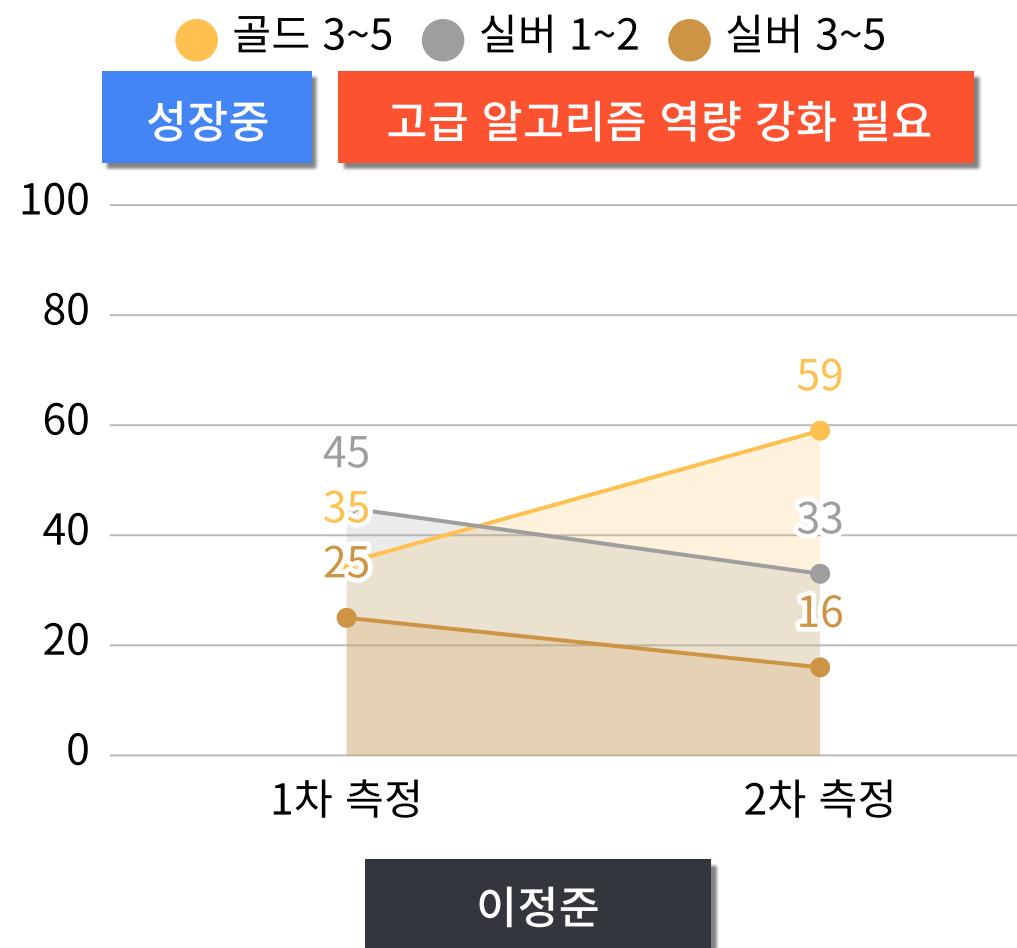
● 골드 3~5 ● 실버 1~2 ● 실버 3~5



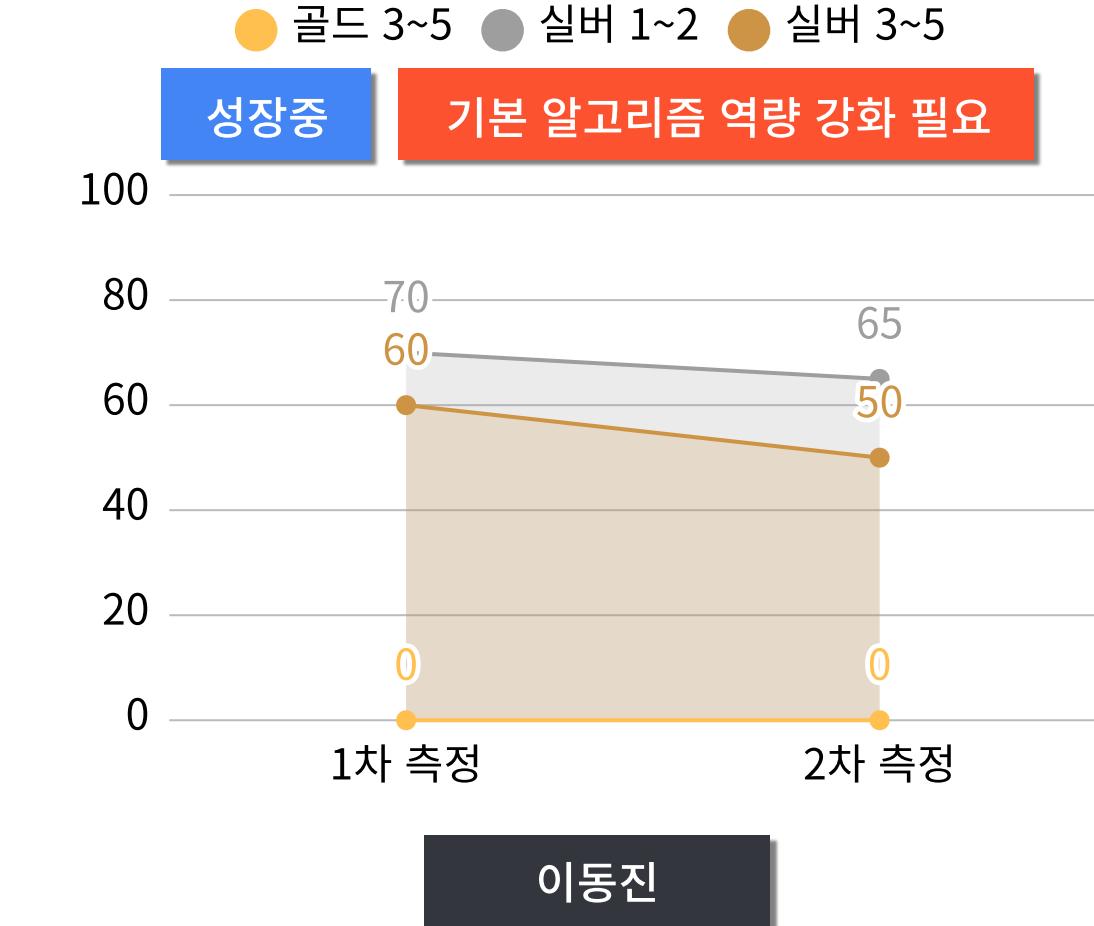
● 골드 3~5 ● 실버 1~2 ● 실버 3~5



● 골드 3~5 ● 실버 1~2 ● 실버 3~5



● 골드 3~5 ● 실버 1~2 ● 실버 3~5



# 제 3장. Computer Science

데이터베이스 및 디자인 패턴

1

매일 공부할 CS 주제를 1~3개 선정한다.

2

정규 시간 혹은 이후에 각자 정해진 주제에 대해 공부한다.

3

다음 날, 사다리를 통해 선정된 발표자는 본인이 공부한 내용을 발표한다.

4

청취자들은 발표자의 내용을 피드백하고  
본인이 추가적으로 더 공부한 내용이 있으면 발표한다.

# 제 3장. Computer Science

## 1~4주차 - 데이터베이스

- ✓ 데이터베이스 풀
- ✓ 정규화(1차, 2차, 3차, BCNF)
- ✓ Join
- ✓ Index
- ✓ RDBMS와 NoSQL
- ✓ 옵티마이저(Optimizer)
- ✓ 파티셔닝
- ✓ Replication
- ✓ 샤딩
- ✓ ORM

## 4~5주차 - 디자인 패턴

- ✓ Singleton 패턴
- ✓ MVC 패턴
- ✓ MVP 패턴
- ✓ MVVM 패턴
- ✓ Factory 패턴
- ✓ Observer 패턴
- ✓ Command 패턴
- ✓ Builder 패턴
- ✓ Adapter 패턴

# 제 3장. Computer Science - 성과

-  **데이터베이스**
  - 1. [데이터베이스] DataBase Connection Pool
  - 2. [데이터베이스] 정규화
  - 3. [데이터베이스] 트랜잭션(Transaction)
  - 4. [데이터베이스] JOIN
  - 5. [데이터베이스] Index
  - 6. [데이터베이스] 효과적인 쿼리
  - 7. [데이터베이스] RDBMS 와 NoSQL
  - 8. [데이터베이스] 옵티마이저
  - 9. [데이터베이스] 파티셔닝
  - 10. [데이터베이스] Replication
  - 11. [데이터베이스] 샤딩
  - 12. [데이터베이스] RDBMS 와 NOSQL
  - 13. [데이터베이스] ORM
-  **디자인 패턴**
  - 1. [디자인 패턴] Singleton 패턴
  - 2. [디자인 패턴] MVC 패턴
  - 3. [디자인 패턴] MVP, MVVM, Flux (feat. React)
  - 4. [디자인 패턴] 랙토리 패턴
  - 5. [디자인 패턴] 음지버 패턴
  - 6. [디자인 패턴] 커맨드 패턴
  - 7. [디자인 패턴] 빌더 패턴
  - 8. [디자인 패턴] 어댑터 패턴
-  **김준서**
  - ▲ 숨기기

## 디자인 패턴

88 글리리 보기 +

### 싱글턴 패턴

- 목적
- 클래스에 인스턴스가 하나만 있도록 하면서 이 인스턴스에 대한 접근 지정을 제공하는 설계 디자인 패턴
- 문제

### 싱글턴 패턴

```

class Singleton {
    private Singleton() {} // Private constructor
    private static Singleton instance = new Singleton();
    public static Singleton getInstance() {
        return instance;
    }
}

```

### MVC

```

class View {
    void update(Model model) {
        // ...
    }
}

class Model {
    void update(View view) {
        // ...
    }
}

class Controller {
    void handle(Event event) {
        Model model = ...
        View view = ...
        model.update(view);
    }
}

```

### 데일러베이스

```

class Address {
    String id;
    String name;
    String street;
    String city;
    String state;
    String zip;
}

class Transaction {
    String id;
    String type;
    String status;
    Date date;
}

class Index {
    String name;
    String type;
    String column;
}

class Database {
    List addresses;
    List transactions;
    List indexes;
}

```

### Observer

```

class Subject {
    void register(Observer observer) {
        observers.add(observer);
    }

    void unregister(Observer observer) {
        observers.remove(observer);
    }

    void notify() {
        for (Observer observer : observers) {
            observer.update();
        }
    }
}

class Observer {
    void update() {
        // ...
    }
}

```

필터
장점
🔍
...
새로 만들기

### 데이터베이스

```

class Address {
    String id;
    String name;
    String street;
    String city;
    String state;
    String zip;
}

class Transaction {
    String id;
    String type;
    String status;
    Date date;
}

class Index {
    String name;
    String type;
    String column;
}

class Database {
    List addresses;
    List transactions;
    List indexes;
}

```

### 트랜잭션 격차 수준

- 포지셔널: 일정 거리를 두고 일정 시간마다 트랜잭션을 처리하는 방식으로 트랜잭션 처리 시간은 일정하지 않다.
- 제작자: 같은 트랜잭션 내에서 일정한 거리를 두고 일정한 시간 간격으로 트랜잭션을 처리하는 방식으로 일정한 거리를 두고 일정한 시간 간격으로 트랜잭션을 처리하는 방식

### JNT(r.id) r\_count...

한승재

## 데이터베이스(Database)

- 1. [데이터베이스] 데이터베이스 풀 (Database Pool)
- 2. [데이터베이스] 정규화(Normalization)란?
- 3. [데이터베이스] 트랜잭션(Transaction)이란?
- 4. [데이터베이스] 트랜잭션 격리 수준(Transaction Isolation Level)
- 5. [데이터베이스] JOIN이란?
- 6. [데이터베이스] INDEX란?
- 7. [데이터베이스] RDBMS와 NoSQL
- 8. [데이터베이스] 효과적인 쿼리 저장
- 9. [데이터베이스] 옵티マイ저(Optimizer)
- 10. [데이터베이스] 파티셔닝(Partitioning)
- 11. [데이터베이스] 사남(Sharding)이란?
- 12. [데이터베이스] 리플리케이션 (Replica)
- 13. [데이터베이스] ORM이란?

▲ 슬기기

## 디자인 패턴(Design Pattern)

- 1. [디자인 패턴] MVC 패턴
- 2. [디자인 패턴] 싱글톤 패턴(Singleton pattern)이란?
- 3. [디자인 패턴] MVP 패턴이란?
- 4. [디자인 패턴] MVVM 패턴이란?
- 5. [디자인 패턴] 팩토리 패턴
- 6. [디자인 패턴] 음지버 패턴(Observer Pattern)
- 7. [디자인 패턴] 커맨드 패턴(Command Pattern)
- 8. [디자인 패턴] 빌더 패턴(Builder Pattern)
- 9. [디자인 패턴] 어댑터 패턴(Adapter Pattern)

▲ 슬기기

조재균

온기종기 DB Study			
Aa	Name	Database	+ ...
	Database Connection Pool	DB	
	데이터베이스 용어 정리	RDBMS	
	スキ마(Schema)	DB	
	Functional dependency	DB	
	정규화(Normalization)	RDBMS	
	트랜잭션(Transaction)이란?	DB	
	트랜잭션 격리 수준(Transaction Isolation Level)	DB	
	Join		
온기종기 디자인 패턴 Study			
Aa	Name	Design Pattern	+ ...
	Index		
	RDBMS와 NoSQL	MVC 패턴	아키텍처
	효과적인 쿼리 저장	Singleton	생성 패턴
	옵티マイ저(Optimizer)	MVP 패턴	아키텍처
	파티셔닝	MVVM 패턴	아키텍처
	샤딩(Sharding)	Observer 패턴	행동 패턴
	리플리케이션(Replication)	Factory 패턴	생성 패턴
	ORM(Object-Relational Mapping)	Command 패턴	행동 패턴
		Builder 패턴	생성 패턴
		Adapter 패턴	구조 패턴

└ database	
└ images	
└ Database_Pool.md	
└ EffeicientQuery.md	
└ Index.md	
└ Join.md	
└ Normalization.md	
└ ORM.md	
└ Optimizer.md	
└ Partitioning.md	
└ Replication.md	
└ SQL&NoSQL.md	
└ Sharding.md	
└ Transaction.md	
└ Transaction_Isolation_Lev	
└ design-pattern	
└ images	
└ Adapter.md	
└ Builder.md	
└ Command.md	
└ FactoryMethod.md	
└ MVC.md	
└ MVP&MVVM.md	
└ Observer.md	
└ Singleton.md	
└ .gitignore	
└ README.md	

- DataBase
  - ① DataBase Pool
  - ② 정규화 1차 2차 3차 BCNF
  - ③ 트랜잭션(Transaction)
  - ④ Join
  - ⑤ Index란
  - ⑥ RDBMS와 NoSQL
  - ⑦ 효과적인 쿼리 저장
  - ⑧ 옵티마이저 (Optimizer)
  - ⑨ 파티셔닝
  - ⑩ Replication
  - ⑪ ⑪ 샤딩(Sharding)
  - ⑫ ⑫ ORM(Object Relational M
- ⚡ 디자인패턴
  - 싱글톤 패턴이란?
  - MVC (Model View Controller) Pattern 이란?
  - MVP(Model View Presenter) Pattern 패턴이란?
  - MVVM(Model View Model View) Pattern이란?
  - + ::‣ Factory Method Pattern
  - Observer Pattern
  - Builder Pattern ( 빌더 패턴 )
  - Adaptor Pattern ( 어댑터패턴 )
  - Command ( 커맨드패턴 )

# 제 4장. Problem Based Learning - 개인

월요일

화요일

수요일

목요일

금요일

토요일

일요일

Computer Science

Problem Based Solved

Problem Solving

CS 리뷰

CS 리뷰

CS 리뷰

CS 리뷰

CS 리뷰

PS 리뷰

PBL 리뷰

# 제 4장. Problem Based Learning - 개인

김준서

프론트엔드 개발자



## Introduce

안녕하세요. 열정, 끈기, 집념의 프론트엔드 개발자 김준서입니다.

저의 개발자 커리어의 목표는

"사용자의 행동 패턴을 수집 후 가공하여 새로운 서비스를 제공하는 시장에서 차별화된 서비스를 제공하는 것입니다."

지금은 프론트엔드 공부에 집중하고 있으며, 백엔드 개발자로도 전환하는 계획입니다. 또한 먼 미래이지만, 데이터 분석에 접근할 수 있는 서비스를 만들고자 합니다.

요즘은 좋은 팀워크를 만들어내기 위해 어떻게

"수평적인 구조여도 팀 내 의사결정자를 정하기",

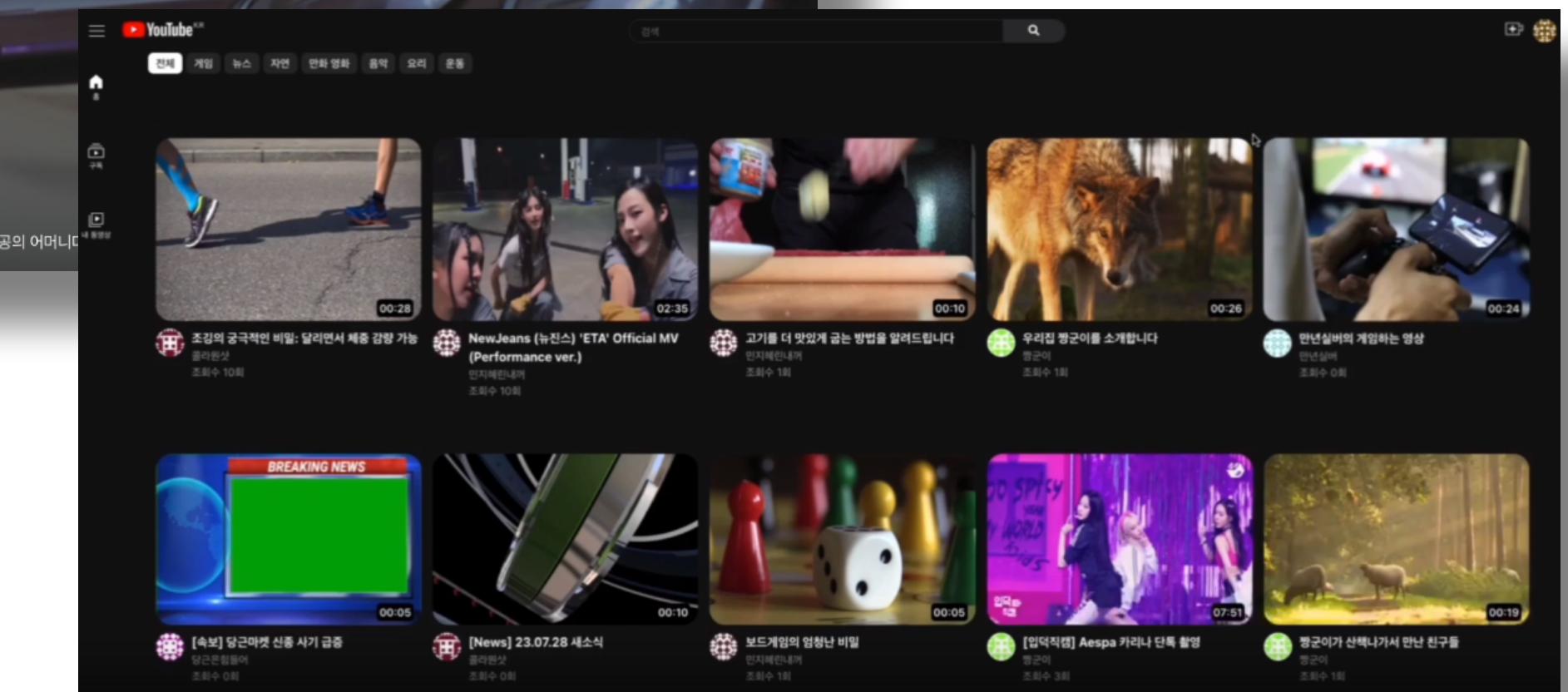
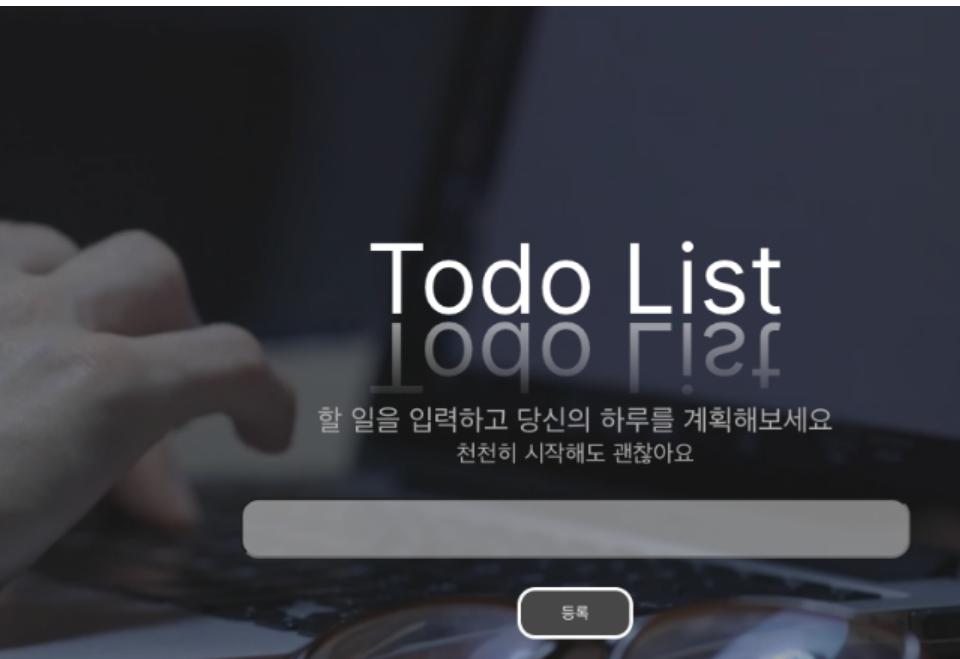
"치명적인 상황이 아니면, 내 의견 애착 버리기" 등 다양한 시도를 해보며 사회적 테크닉 증진에 노력을 기하고 있습니다.

## Skills

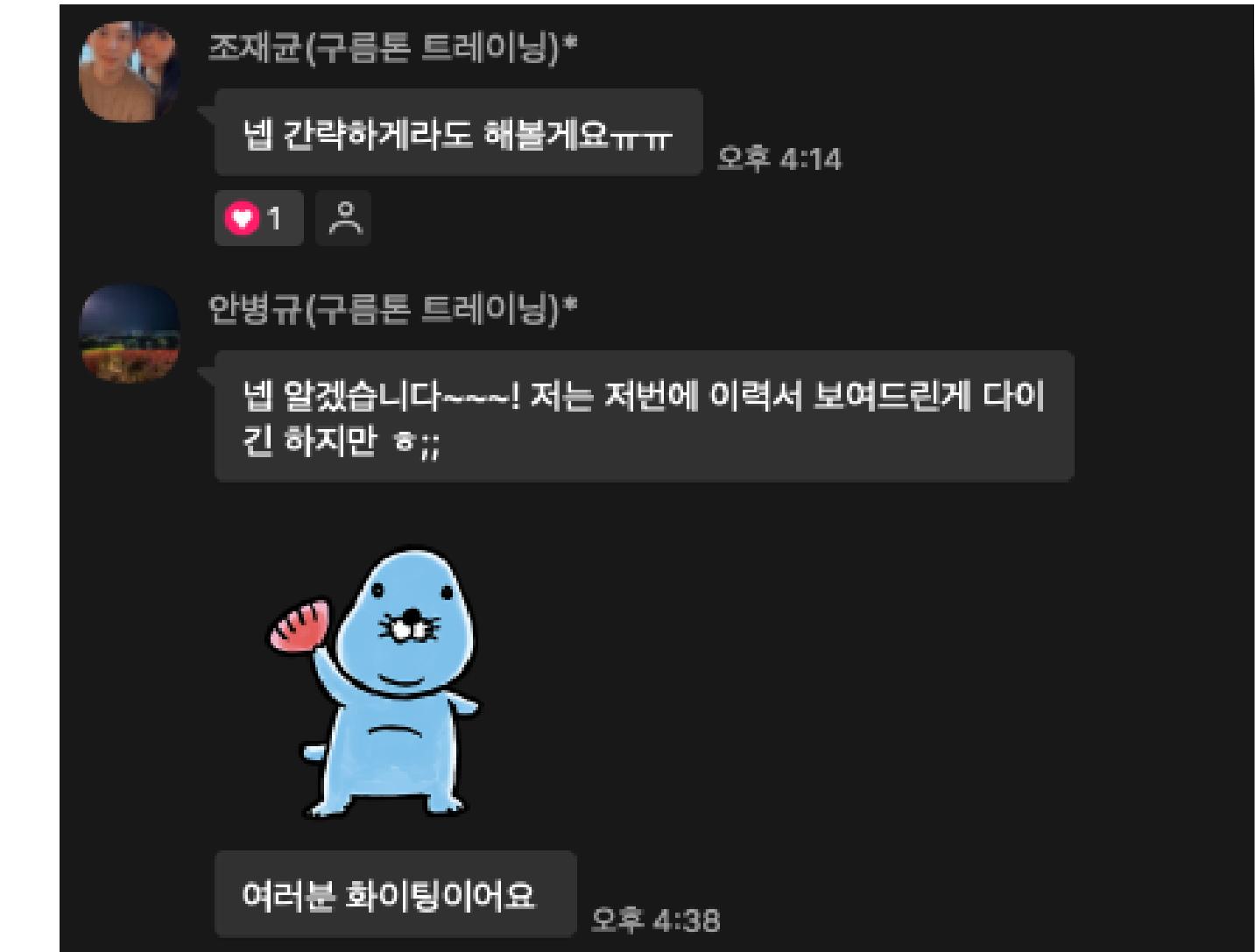
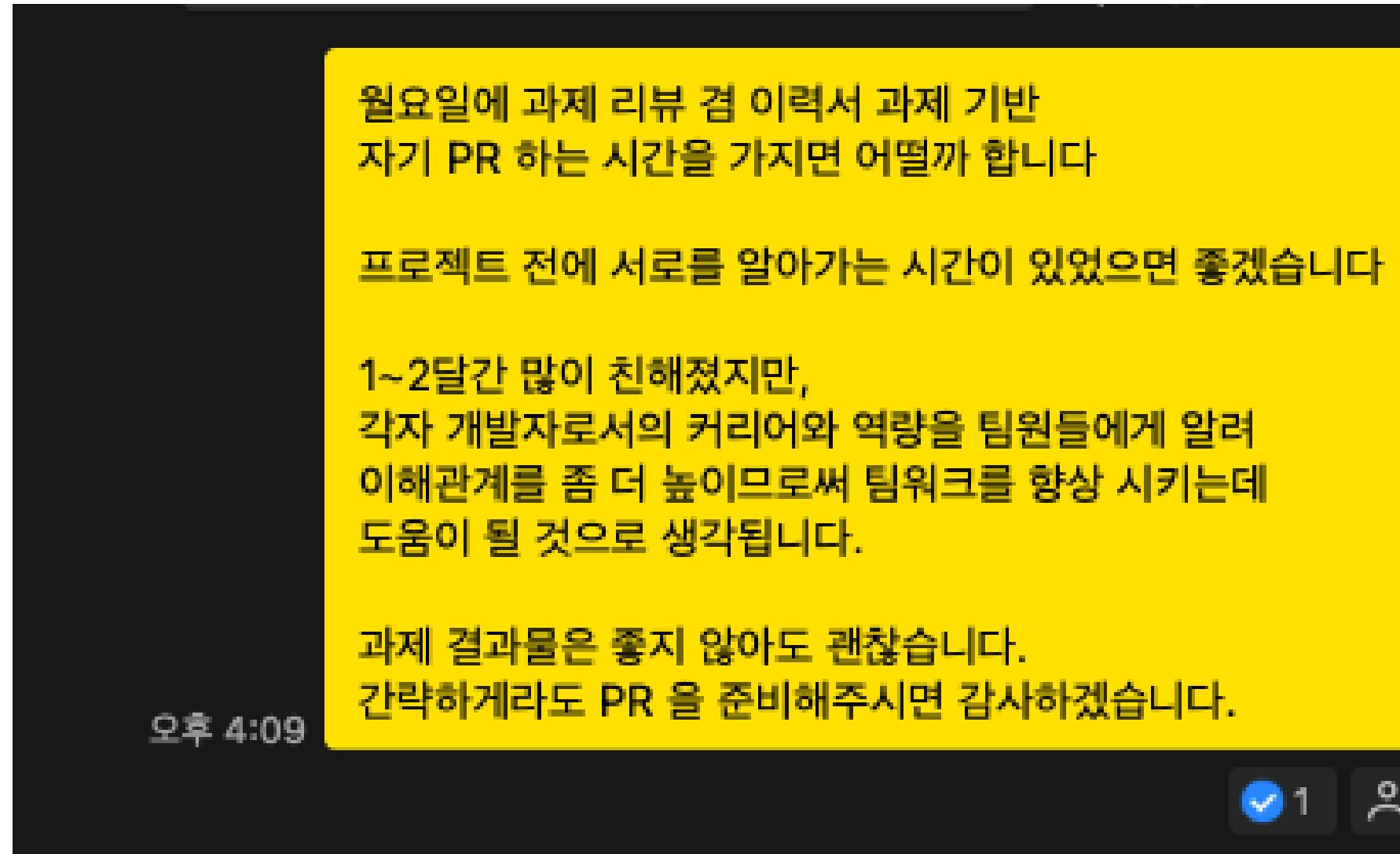
### Frontend

React  
React Native

### ETC



# 제 4장. Problem Based Learning - 개인



## [팀프로젝트 대비]

1. 팀원들 간의 역량을 서로 파악하기 위함
2. 팀장으로서 역량에 따른 역할 위임을 하기 위함

# 제 4장. Problem Based Learning - 팀

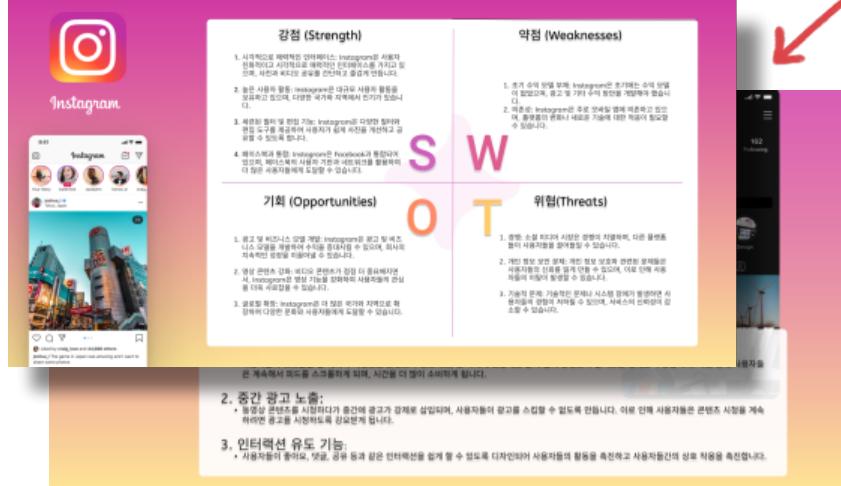


Instagram



OGTOUR  
9oorm travel

- 인스타 스토리 조회 내부 들어 보았을 때
  - 조회할 수 있는 모든 스토리 데이터를 한번에 받아오는 것이 아니다.
  - 한번에 7개의 스토리 데이터를 받아온다.
- 실제 스토리 데이터는 5개이며 왼쪽, 오른쪽 영역에 더미 스토리 데이터가 2개 더 존재한다.
- 현재 재생되고 있는 스토리 외에는 썸네일로 되어 있는 것 같고, 실제 보고 있는 스토리만 미디어 데이터를 받아온다.
- 스토리를 이동하게 되면 재생되는 스토리를 기준으로 9개의 스토리가 동적으로 변경되는 것으로 보인다.
  - 즉, 정확한 내부 로직은 알 수 없으나 스토리를 이동할 때 다음 스토리의 정보를 다시 매핑하는 것으로 보인다.
- 스토리는 24시간 뒤에 사라진다.
- 스토리는 수정이 불가능하다.



	TypeScript
• 버전	
• 사용 이유	
◦ 타입을 통해 코드의 안정성을 높이기 위해	
	리액트
• 버전	
• 사용 이유	
◦ SPA 구현	
	recoil
• 버전	
• 사용 이유	
◦ 직관적이면서 간단해 진입 장벽 낮음	
◦ 빠른 시간 안에 완성해야 하므로 간단한 구조인 리코일 채택	
	styled-components
• 버전	
• 사용 이유	
◦ 모두가 사용해본 기술이라 짧은 시간 안에 결과를 내기에 적합하다고 판단	
	Guitar 라이브러리
• react-router-dom 버전	
◦ <a href="https://forj.tistory.com/160">https://forj.tistory.com/160</a>	
• react-icons 버전 - 4.10.1	
• react-responsive 버전 - 9.0.2	
• react-dropzone 버전 - 14.2.3	
• axios 버전 - 1.4.0	
• eslint	
• prettier	

프로덕트 매니저 - 김준서  
의사결정권자 - PM

## 프론트엔트 및 백엔드 팀장 배정

### Instagram 기능 분석 및 구현 기능 선정

프론트엔드 팀장 - 한승재  
팀원 - 김준서, 조재균

### PBL 과제 제출물 제작

### 페이지 별 필요 API 정리

### 통합 회의 - API 목록 보완

의사결정권자 - 프론트엔드 팀장

### 협업 규칙 정의

### 기술 스택 결정 및 보일러플레이트 제작

### 페이지별 개발 역할 분담

### Github Pages 배포

### 여행 상품 판매 사이트 기획 및 디자인

### 통합 테스트

### 알파 테스트

백엔드 팀장 - 안병규  
팀원 - 이정준, 이동진

### 엔티티 별 속성 정리

### ERD 제작

의사결정권자 - 백엔드 팀장

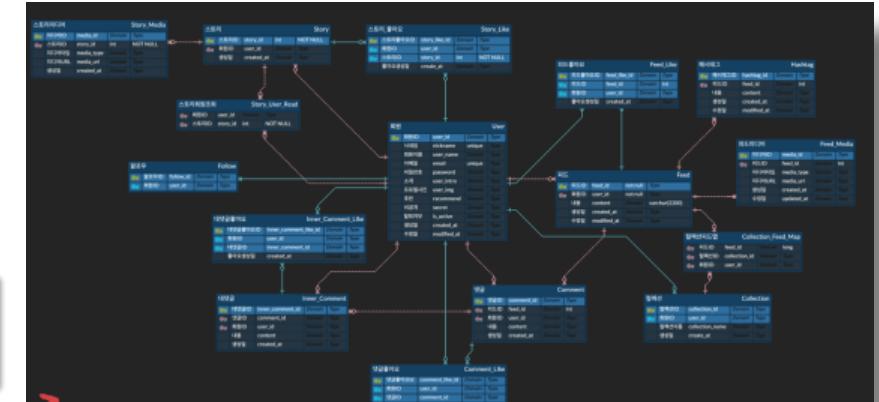
### 협업 규칙 정의

### 기술 스택 결정

### 기능별 개발 역할 분담

### AWS 인스턴스 배포 및 Docker 구축

### Docker CI/CD 에러 해결



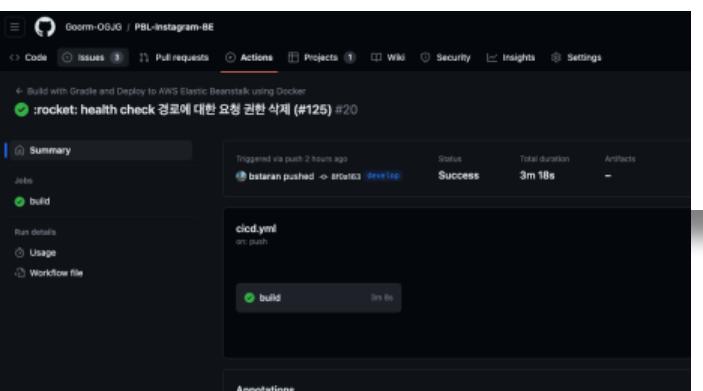
## Back

- Java 17
- Spring
- Spring Security
- Spring Boot 3.1.2
- JPA
- Gradle
- JUnit5

- Logback
- h2
- ▶ MySQL

## Infra

- AWS S3



# 제 4장. Problem Based Learning

## - 여행 상품 판매 사이트

The homepage features a large banner with a dark blue and white marbled background. In the center, there's a title '제주 여행 패키지' (Jeju Travel Package) and a subtitle '내가 원하는 일정에 가능한 항공, 숙박, 렌터카를 할인된 가격으로 한번에!!'. Below this are four package options: '항공+숙박+렌터카 출인원 패키지', '항공+숙박 스카이 레지던스 패키지', '항공+렌터카 스카이 드라이브 패키지', and '숙박+렌터카 컴포트 드라이브 패키지'. At the bottom, there are input fields for '출발일' (Departure Date: 2023-08-28), '종료일' (End Date: 2023-08-31), '인원' (Number of People: 성인 3인, 소아 0인), and a large blue button '패키지 예약하기' (Book Package). The footer contains contact information (고객센터 1544-3100), operating hours (09:00-18:00), and a copyright notice (Copyright © OGU.COM, All rights reserved.).

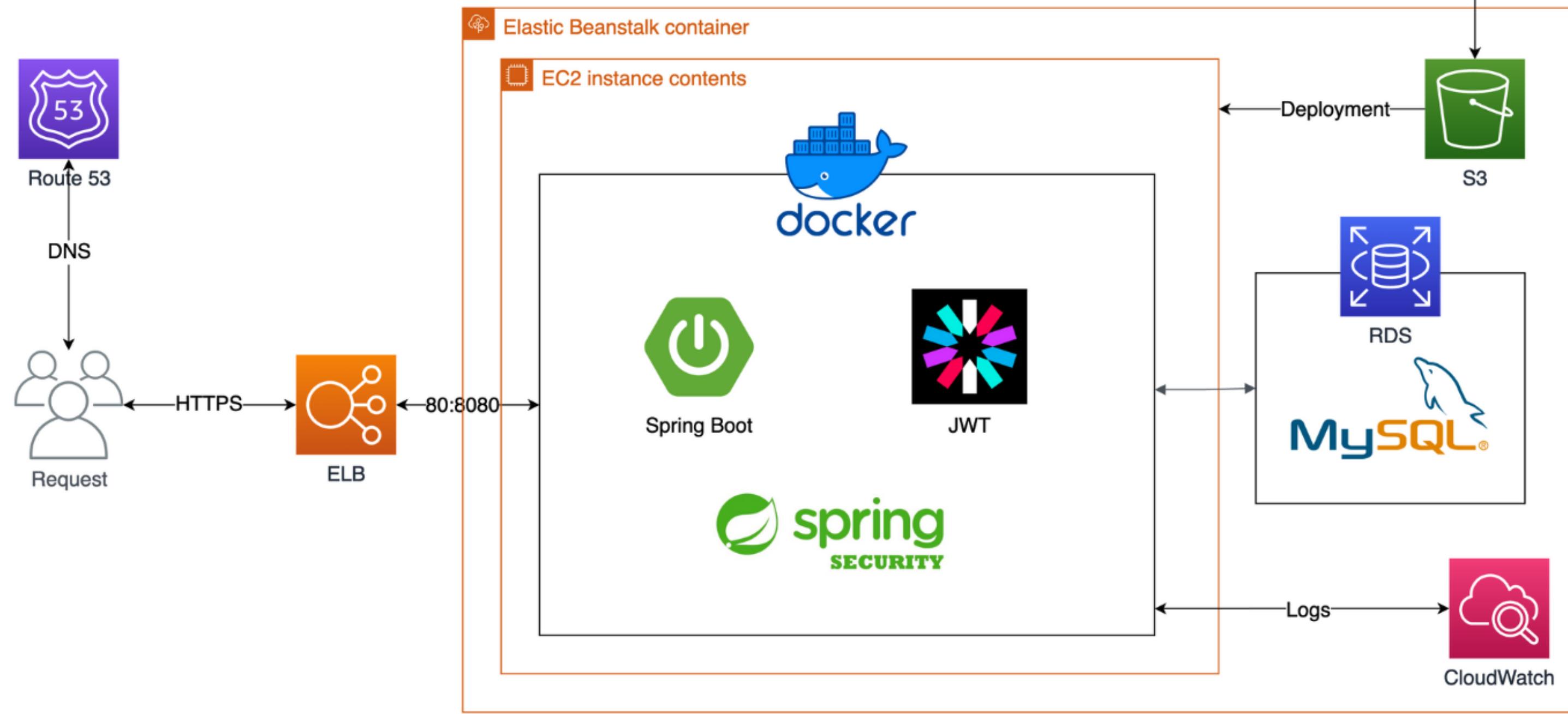
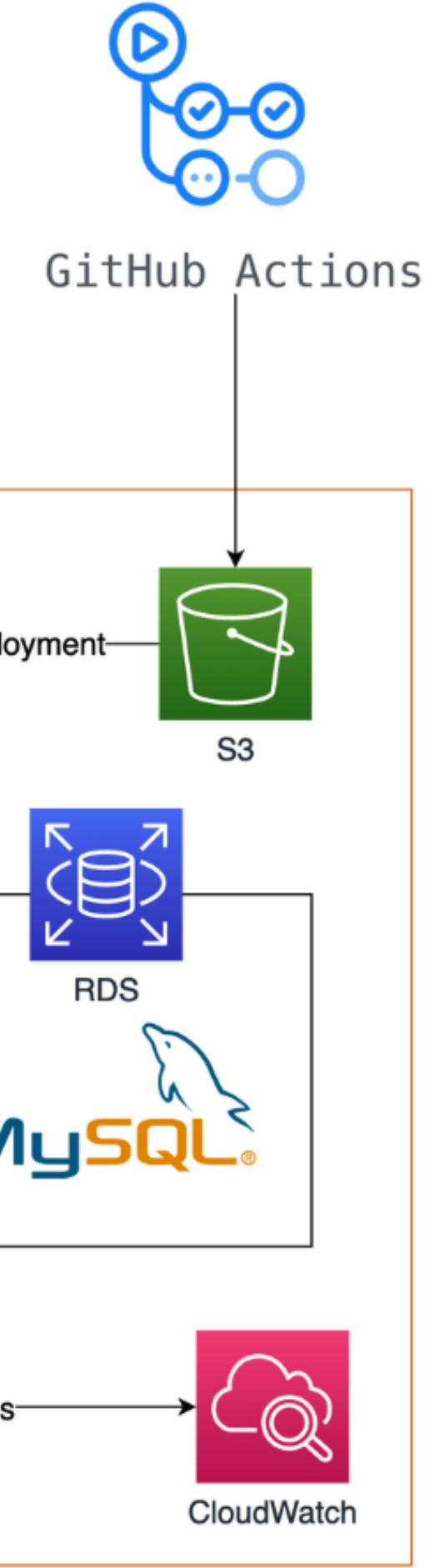
The booking process consists of several steps:  
1. \*\*Booking Confirmation\*\*: Shows a message from OGTour confirming the booking for 08월 28일 (월) to 08월 29일 (화).  
2. \*\*Flight Selection\*\*: Displays flight options from Gimpo (GMP) to Jeju (CJU) on 06:15 and 07:30. A summary shows a total airfare of 80,600 won.  
3. \*\*Accommodation Selection\*\*: Shows a hotel room with a double bed and a nightstand. A summary shows a total accommodation cost of 245,900 won.  
4. \*\*Car Selection\*\*: Shows a car interior. A summary shows a total car rental cost of 81,700 won.  
5. \*\*Payment Confirmation\*\*: Shows a summary of the total payment amount: 1,314,500 won, with breakdowns for airfare, accommodation, and car rental. A red '결제하기' (Pay) button is at the bottom.

# 제 4장. Problem Based Learning

## - 여행 상품 판매 사이트



# 제 4장. Problem Based Learning - Instagram



# 제 4장. Problem Based Learning - FE

## Font

### Font Size

폰트 사이즈 명	폰트 크기
XS	12px
S	14px
M	16px
L	24px
XL	30px

#### 사용 아이콘 정리

- 아이콘
  - 인스타 로고 아이콘 - BsInstagram
  - 홈버튼 - GoHome / GoHomeFill
  - 검색버튼 - GoSearch
  - 만들기 버튼(+) - RiAddBoxLine / RiAddBoxFill
  - 햄버거 버튼(메뉴) - AiOutlineMenu
  - 하트 - AiOutlineHeart / AiFillHeart
  - 글(말풍선) - FaRegComment / FaComment
  - 그리드 (게시판) - BsGrid3X3 / MdGridOn
  - 저장됨 아이콘 - BsBookmark / BsBookmarkFill
  - 가로 땡땡땡 - HiEllipsisHorizontal / HiDotsHorizontal(뚜꺼비)
  - 동그라미 플러스 - TbCirclePlus
  - X표시 - GrClose
  - 플레이 / 멈춤 - HiPlay / HiPause
  - 소리 끄고 / 키는거 - ImVolumeMedium / ImVolumeMute2
  - 캐리셀 좌우 버튼 <, > - CiCircleChevLeft / CiCircleChevRight
  - 피드 사진여러개 - PiCards, TbBoxMultiple

#### 로그인 문제발생시

- 좌물쇠 - GoLock
- 우물쇠 - CiLock

### Font Weight

폰트 굵기 명	폰트 굵기
Medium	400
Bold	600

## Git 컨벤션

git commit -m "title" -m "body"

Feat : UI/기능 추가 및 수정

- 💡 Feat: [컴포넌트] UI 구현 완료
- ✨ Feat: [기능 이름] 기능 구현/수정 완료
- 🔥 Feat: [기능 이름] 기능 삭제 완료

Fix: 버그 처리

- ▶ Github Issue - 버그 발견 즉시 Github Issue에 버그 리포팅
- 👉 Fix: #Issue1 [Issue 제목] 해결

Docs: 문서(Readme.md 등) 수정

- 📚 Docs: [파일명] [수정사항]

Style: 코드 포맷팅(개행, 인데нт 등, 세미콜론 누락) 등 - UI/기능 변동사항 없음

- 🎨 Style: [파일명] [수정사항]

Refactor: 코드 리팩토링(기능 핫스왑 등)

- 👉 Refactor: [파일명] [리팩토링 간단설명]

Perf: 성능 향상

- ↗️ Perf: [성능 향상 내역]

Chore: 빌드 업무 수정, 패키지 매니저(package.json) 수정

- 🚀 Chore: [라이브러리명] 추가

## Color

### Gray

색상 명	색상 코드	용도
Gray1	#F5F5F5	유저 아이디, 모두 보기 등
Gray2	#A8A8A8	00님이 팔로우 합니다. 댓글 모두보기 등
Gray3	#737373	footer
Gray4	#363636	어두운 버튼

### Blue

색상 명	색상 코드	용도
Blue1	#E0F1FF	태그, @유저 닉네임
Blue2	#0095F6	버튼 배경, 글씨 색

### Red

색상 명	색상 코드	용도
Red1	#ED4956	취소, 신고

## 그래디언트

```
background: #833ab4; /* fallback for old browsers */  
background: -webkit-linear-gradient(to bottom, #fcb045, #fd1d1d, #833ab4); /* Chrome 10-12.0 */  
background: linear-gradient(to bottom, #fcb045, #fd1d1d, #833ab4); /* W3C, IE 10+ */
```

## color.ts

```
export const Blue1 = "#E0F1FF";
export const Blue2 = "#0095F6";
export const Blue3 = "#003366";
```

## font.ts

```
export const XS = "12px";
export const S = "14px";
export const M = "16px";
```

## Login.tsx

```
import styled from "styled-components";
import * as COLOR from "../../constants/color";
import * as FONT from "../../constants/font";
import { Link } from "react-router-dom";
```

```
export const LoginButton = styled.button<{ disabled: boolean }>`  
  color: ${COLOR.White};  
  width: 100%;  
  height: 38px;  
  background-color: ${COLOR.Blue2};  
  border: none;  
  border-radius: 10px;  
  font-size: ${FONT.S};  
  font-weight: ${FONT.Bold};  
  margin-top: 25px;  
  cursor: pointer;  
  
  &:disabled {  
    background-color: ${COLOR.ButtonDisabled};  
    cursor: auto;  
  }  
`;
```

# icon.tsx

```
import { BsBookmark, BsBookmarkFill, BsGrid3X3, BsInstagram, BsAt } from "react-icons/bs";
import { GoHome, GoHomeFill, GoLock, GoSearch } from "react-icons/go";
import { RiAddBoxLine, RiAddBoxFill } from "react-icons/ri";
import { AiOutlineMenu, AiOutlineHeart, AiFillHeart } from "react-icons/ai";
import { FaComment, FaPause, FaPlay, FaRegComment } from "react-icons/fa";
import { HiEllipsisHorizontal } from "react-icons/hi2";
import { HiDotsHorizontal } from "react-icons/hi";
import { TbCirclePlus, TbBoxMultiple, TbExchange } from "react-icons/tb";
import { IoClose } from "react-icons/io5";
import { ImVolumeMedium, ImVolumeMute2 } from "react-icons/im";
import { BiSolidChevronLeftCircle, BiSolidChevronRightCircle, BiMoviePlay } from "react-icons/bi";
import { PiImagesSquareThin } from "react-icons/pi";

interface IconType {
  size?: number;
  color?: string;
}

// 로고
export const Insta = ({ size, color }: IconType) => (
  <BsInstagram size={size} color={color} />
);

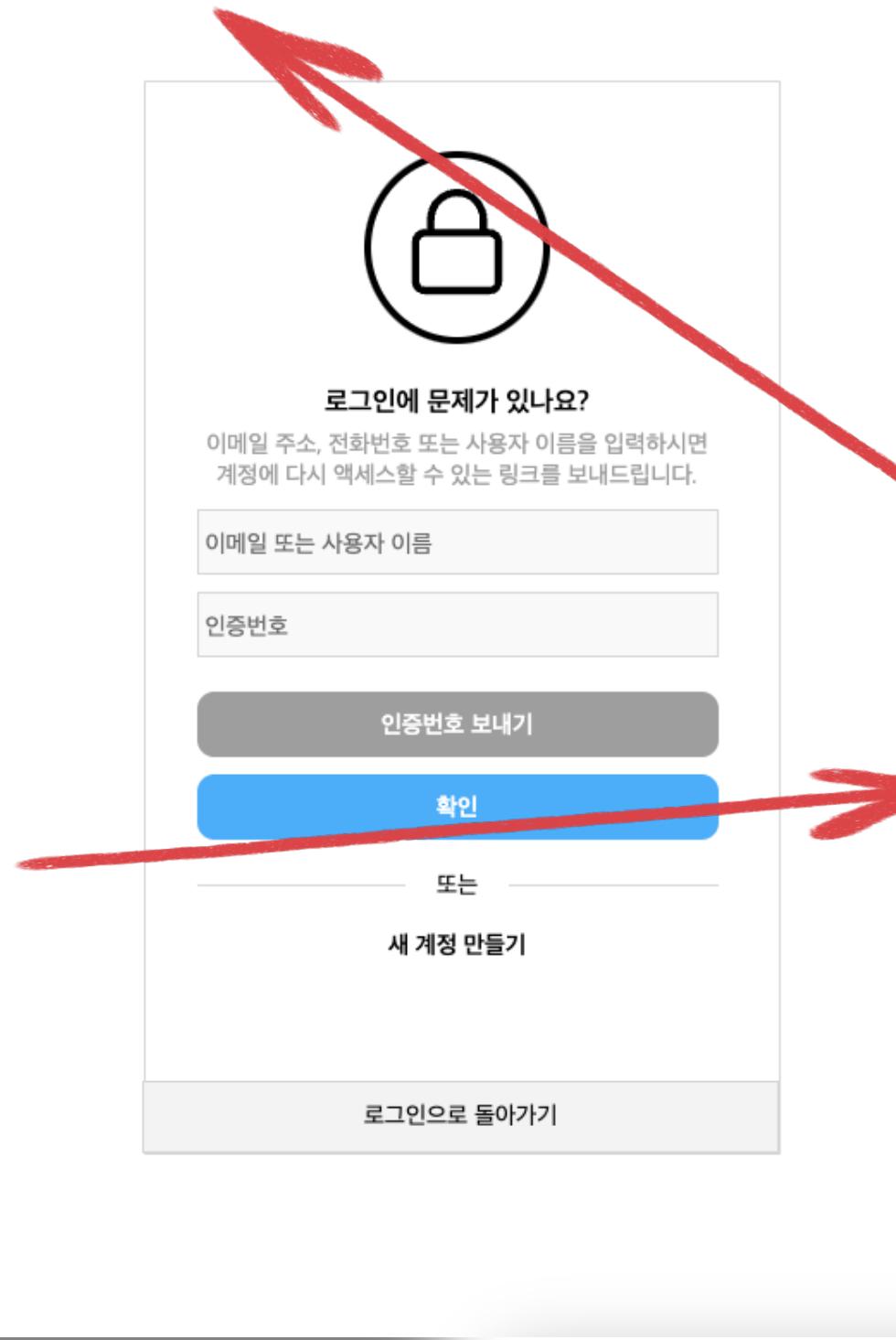
export const InstaLogo = ({ size }: IconType) => (
  <img src={`/images/logoIcon.png`} width={size}></img>
);

export const InstaTextBlack = ({ size }: IconType) => (
  <img src={`/images/logoText_black.png`} width={size}></img>
);

export const InstaText = ({ size }: IconType) => (
  <img src={`/images/logoText.png`} width={size}></img>
);

// 홈
export const Home = ({ size, color }: IconType) => <GoHome size={size} color={color} />;
export const HomeFill = ({ size, color }: IconType) => (
  <GoHomeFill size={size} color={color} />
);
```

Instagram



# FindPassword.tsx

```
import { BsAt, BsGrid3X3, BsInstagram, BsText } from "react-icons/bs";
import { GoSearch } from "react-icons/go";
import { RiAddBoxLine, RiAddBoxFill } from "react-icons/ri";
import { AiOutlineMenu, AiOutlineHeart, AiFillHeart } from "react-icons/ai";
import { FaComment, FaPause, FaPlay, FaRegComment } from "react-icons/fa";
import { HiEllipsisHorizontal } from "react-icons/hi2";
import { HiDotsHorizontal } from "react-icons/hi";
import { TbCirclePlus, TbBoxMultiple, TbExchange } from "react-icons/tb";
import { IoClose } from "react-icons/io5";
import { ImVolumeMedium, ImVolumeMute2 } from "react-icons/im";
import { BiSolidChevronLeftCircle, BiSolidChevronRightCircle, BiMoviePlay } from "react-icons/bi";
import { PiImagesSquareThin } from "react-icons/pi";

import { InstaTextBlack, Lock } from "../../../../components/icon";
import { InputBox } from "../../../../components/inputBox";
```

```
return (
  <S.Container>
    <S.Nav>
      <InstaTextBlack size={100} />
```

# 제 4장. Problem Based Learning - BE

노션을 활용한 프로젝트 페이지를 생성하고 문서를 공유

Gallery

## Back-End

- [Doc] 개발 문서
- 모델링
- 기술 스택

- 컨벤션
- 트러블슈팅
- 업무 분담

레퍼런스

### Task 관리

- ▶ Task 진행 순서
- ▶ Github project를 통해 관리
- ▶ issue 만들기
- ▶ Pull Requests(중요)

### Git commit 커번션

- ▶ Rule

### 코드 커번션

- ▶ Code

Table +

Filter Sort ⚡ Q ↻ ... New

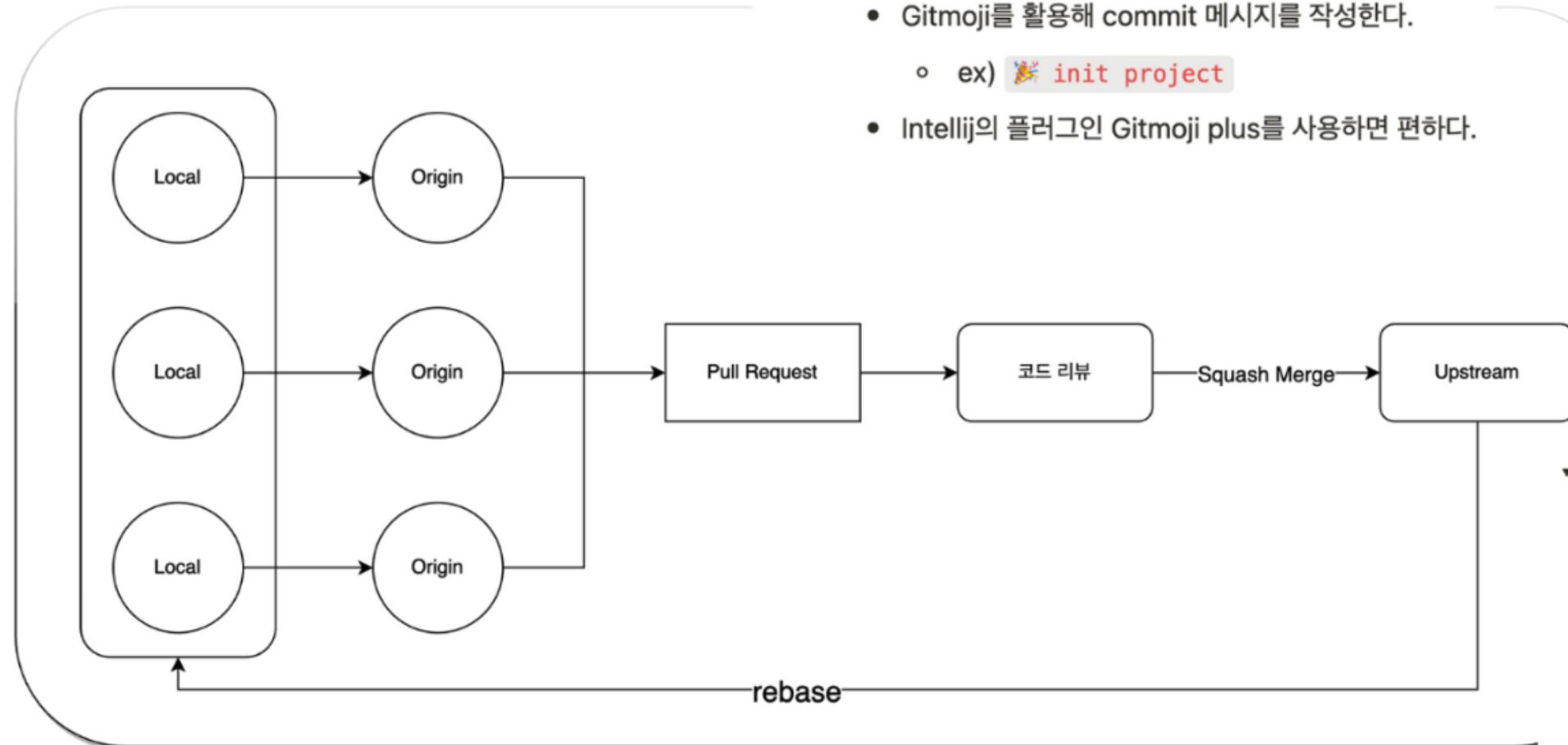
▶ 백엔드 문서 ...

Type	Aa Name	Project	Date
설정 관련	MySQL DB TABLE CREATE QUERY	인스타그램	August 7, 2023
설정 관련	JWT Secret Key	인스타그램	
설정 관련	Application.yaml 예시	인스타그램	August 7, 2023
사용 방법	git rebase	인스타그램	
문제 해결	security h2 오류	인스타그램	
사용 방법	JWT 활용 참고	인스타그램	
문제 해결	cors Error	인스타그램	
설정 관련	mail sender 설정	인스타그램	
기능 구현	Global Exception Handling	인스타그램	
리서치	개발 환경	인스타그램	August 30, 2023
사용 방법	JIRA 사용 커번션	JIRA	August 29, 2023
설정 관련	DB TABLE QUERY	OGTour	August 27, 2023
사용 방법	API 문서 설명 가이드	OGTour	August 27, 2023
리서치	API 설계 & 툴	OGTour	August 26, 2023
설정 관련	Logging 전략	인스타그램	September 1, 2023
	프로젝트 마무리 회고	인스타그램	

+ New Calculate COUNT 16

# 제 4장. Problem Based Learning - BE

## Git 협업 전략



### ▼ Rule

- Gitmoji를 활용해 commit 메시지를 작성한다.
  - ex) 🎉 init project
- IntelliJ의 플러그인 Gitmoji plus를 사용하면 편하다.

### 코드 컨벤션

#### ▼ Code

- 상수(final) : 대문자
- 변수 : 소문자 or 카멜
- 메소드 : 카멜
- 클래스 : 파스칼
- URL : 소문자 + 명사(자원) + 언더바(\_) 대시[-]로
- 변수키워드 정렬 순서 : private static final (PSF)
- 생성자 빌더 사용

! 소트워크 앤솔루지 - 객체 지향 프로그래밍을 잘하기 위한 9가지 원칙을 제시하고 있다.

- ▶ 1. 한 메서드에 오직 최대 3단계의 들어쓰기만 한다.
- ▶ 2. else 예약어를 쓰지 않는다.
- ▶ 3. 모든 원시 값과 문자열을 포장한다. (래핑)
- ▶ 4. 한 줄에 점을 하나만 찍는다.
- ▶ 5. 줄여 쓰지 않는다(축약 금지).
- ▶ 6. 모든 엔티티를 작게 유지한다.
- ▶ 7. 3개 이상의 인스턴스 변수를 가진 클래스를 쓰지 않는다.
- ▶ 8. 일급 컬렉션을 쓴다.
- ▶ 9.-getter/setter/프로퍼티를 쓰지 않는다.

다소 깐깐한 컨벤션을 적용했지만,  
이로 인해 프로젝트를 효율적으로 관리할 수 있었다.

# 제 4장. Problem Based Learning

## - Instagram

- ✓ 로그인
- ✓ 로그아웃
- ✓ AccessToken 재발급
- ✓ 회원가입
- ✓ 이메일 중복 체크
- ✓ 닉네임 중복 체크
- ✓ 이메일을 통한 인증번호 전송
- ✓ 인증번호 일치 여부 확인
- ✓ 프로필 가져오기
- ✓ 내 피드 목록 가져오기

- ✓ 내 피드 목록 가져오기 - 무한 스크롤 9개씩
- ✓ 내가 보관한 피드 목록 가져오기 - 무한 스크롤 9개씩
- ✓ 프로필 수정 페이지 정보 가져오기
- ✓ 프로필 수정하기
- ✓ 프로필 이미지 수정하기
- ✓ 팔로우 하기
- ✓ 언팔로우 하기
- ✓ 내가 팔로우한 사용자 목록 불러오기
- ✓ 나를 팔로우한 사용자 목록 불러오기

- ✓ 아이디 검색
- ✓ 태그 검색
- ✓ 태그 정보 및 태그 걸린 피드 가져오기
- ✓ 내가 팔로우한 사람들이 올린 스토리 목록 가져오기
- ✓ 스토리 작성하기
- ✓ 스토리 읽음 처리
- ✓ 스토리 삭제
- ✓ 스토리 좋아요
- ✓ 스토리 좋아요 취소

- ✓ 피드 목록 가져오기
- ✓ 피드 상세 정보 불러오기
- ✓ 좋아요 누른 최근 사용자 100명 가져오기
- ✓ 피드 대댓글 정보 가져오기
- ✓ 피드 작성하기
- ✓ 피드 삭제
- ✓ 피드 보관함 추가
- ✓ 피드 보관함 삭제
- ✓ 피드 좋아요
- ✓ 피드 좋아요 취소
- ✓ 피드 좋아요 누른 사용자 목록 가져오기

- ✓ 피드 댓글 작성
- ✓ 피드 댓글 삭제
- ✓ 피드 댓글 좋아요
- ✓ 피드 댓글 좋아요 삭제
- ✓ 피드 댓글 좋아요 누른 사용자 목록 가져오기
- ✓ 피드 대댓글 작성하기
- ✓ 피드 대댓글 삭제
- ✓ 피드 대댓글 좋아요
- ✓ 피드 대댓글 좋아요 삭제
- ✓ 피드 대댓글 좋아요 누른 사용자 목록 가져오기

49개의 API 구현 완료



OGJG-Instagram

# **제 4장. Problem Based Learning**

## **- Instagram**

간단 시연

# 제 5장. 스터디 회고

## 좋았던 점

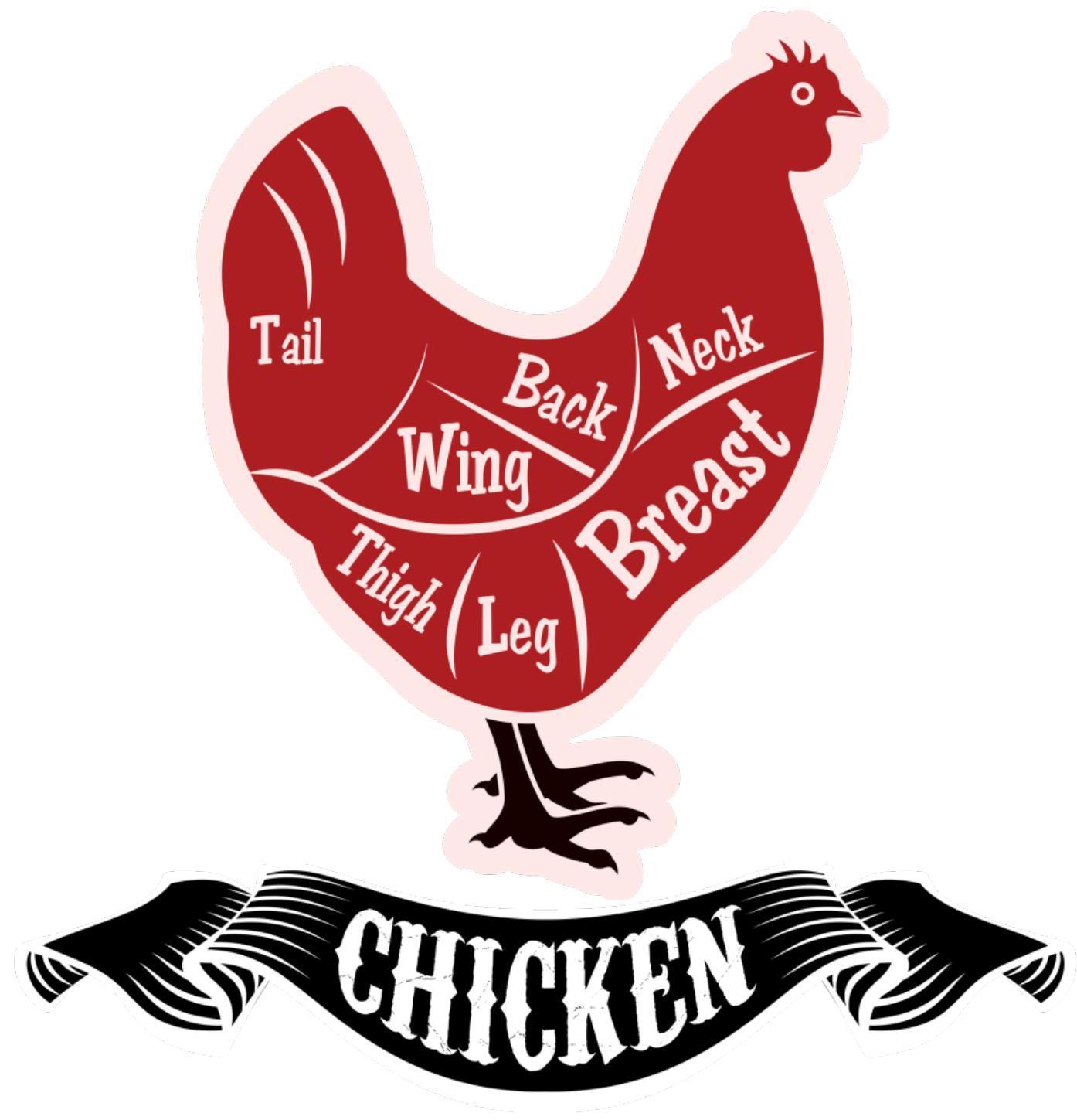
백준 기준 실버 1~2 나이도 문제를 30분 안에 풀 수 있도록 알고리즘 문제 풀이 역량 향상

## 아쉬웠던 점

기술 면접을 대비해 CS 기초 지식을 습득하고 공부한 지식을 모아 각자 CS 문서를 생성

## 종합

개인 PBL 과제를 통해 역량을 쌓고,  
팀 PBL 과제를 하면서 기업 연계 프로젝트에 대비하기 위한 Git 전략, 코딩 컨벤션 등의 협업 규칙들을 정하여 숙달 및 문서화



감사합니다