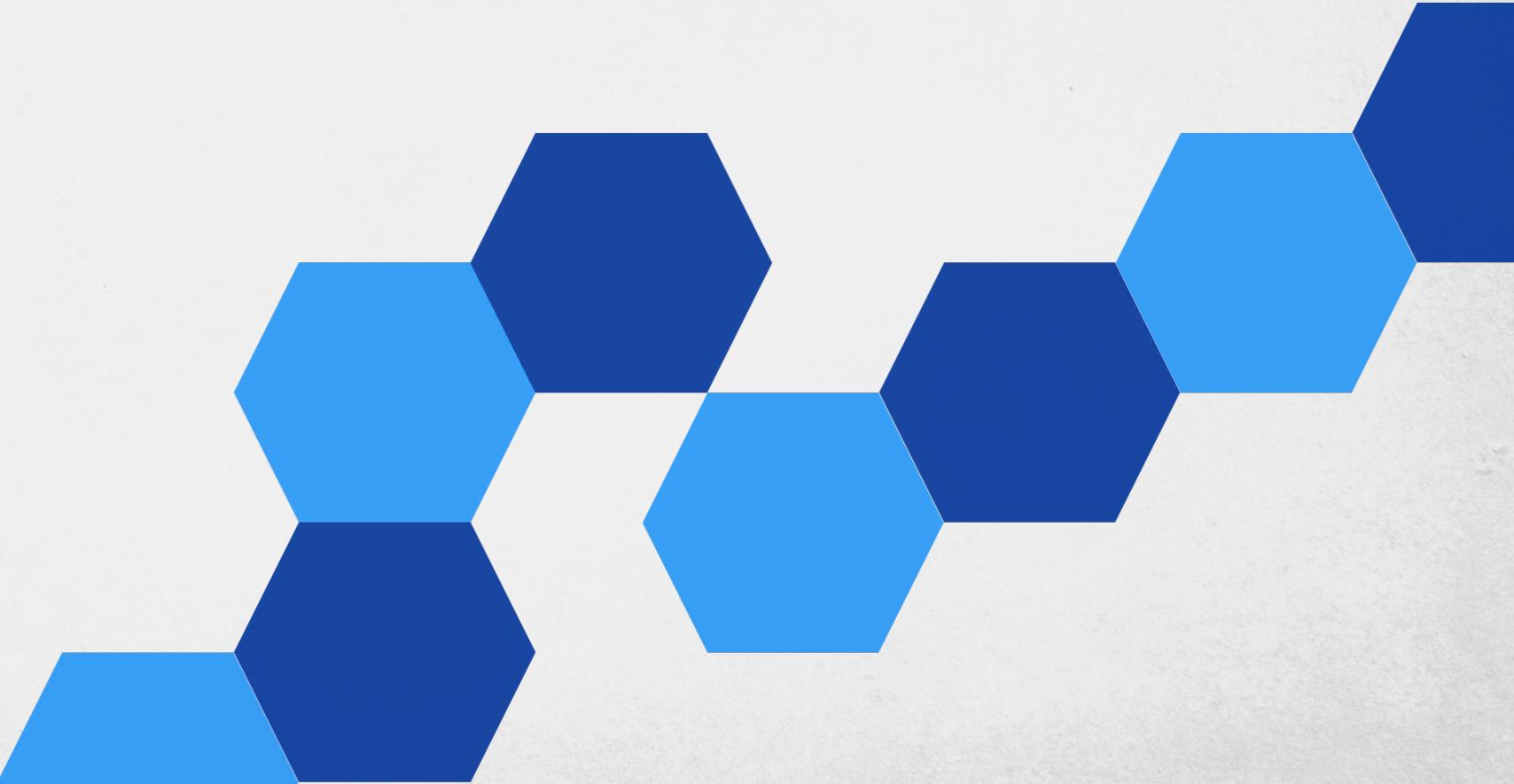


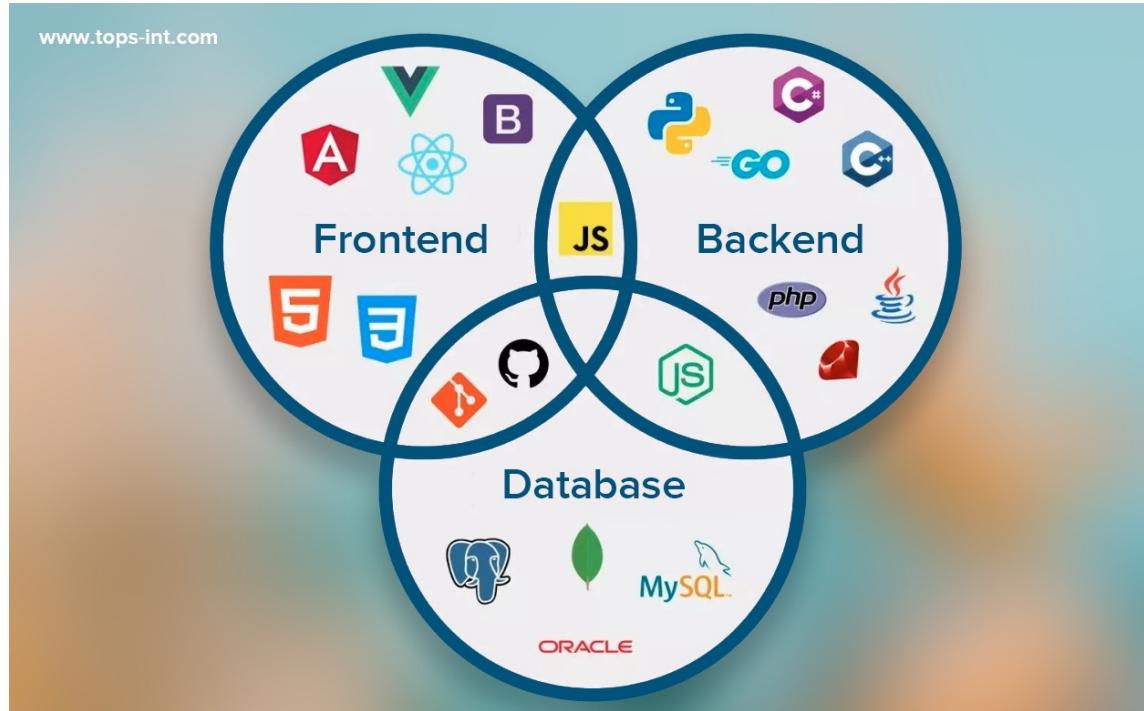
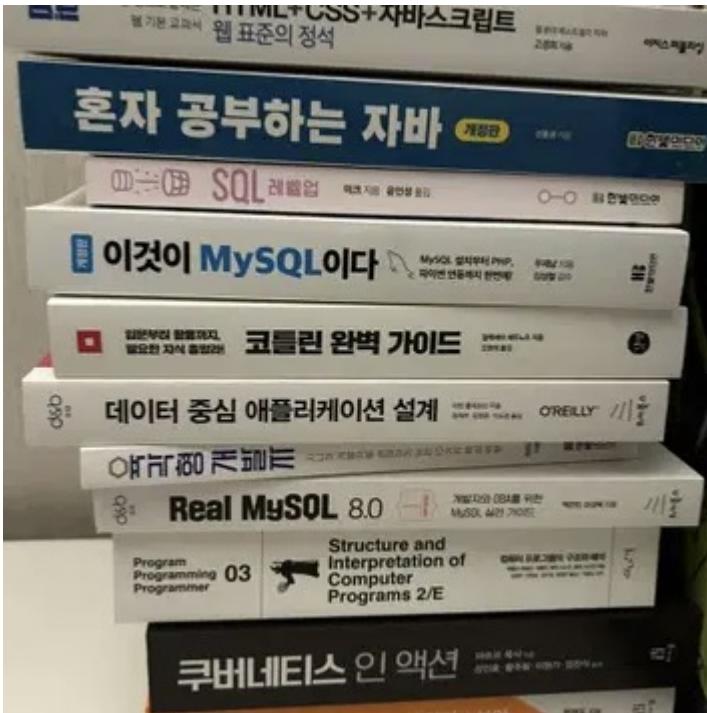
WebShooter

소프트웨어공학개론 1조

권서진, 김문수, 김현진, 박경일, 이유진



이론 위주의 학습

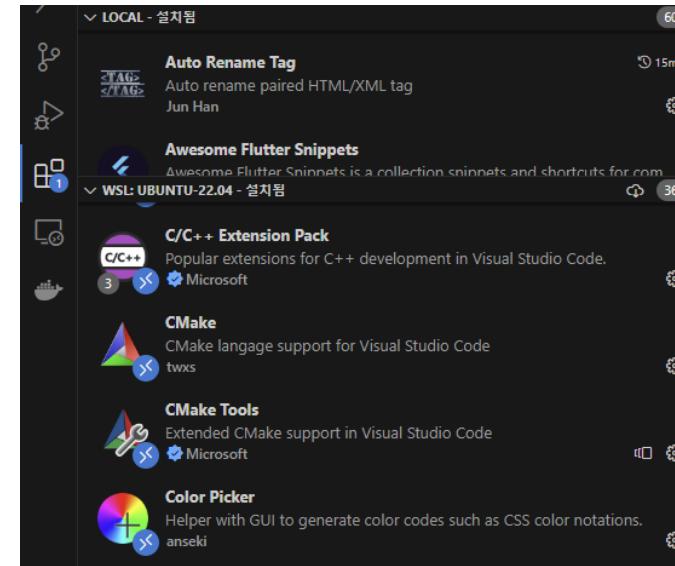
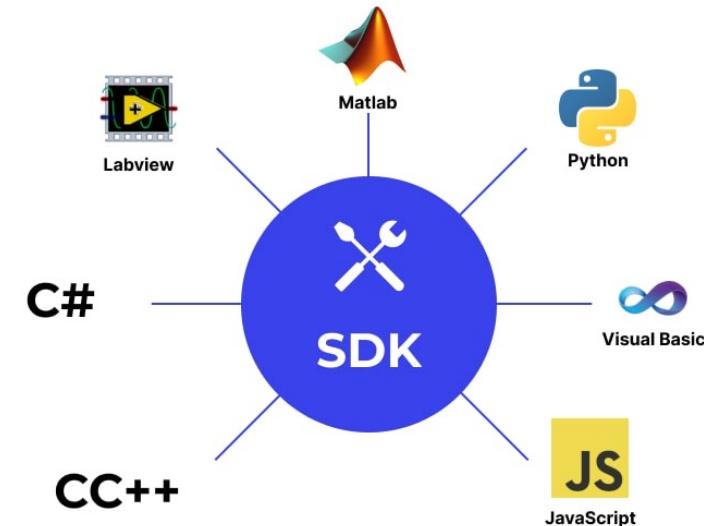


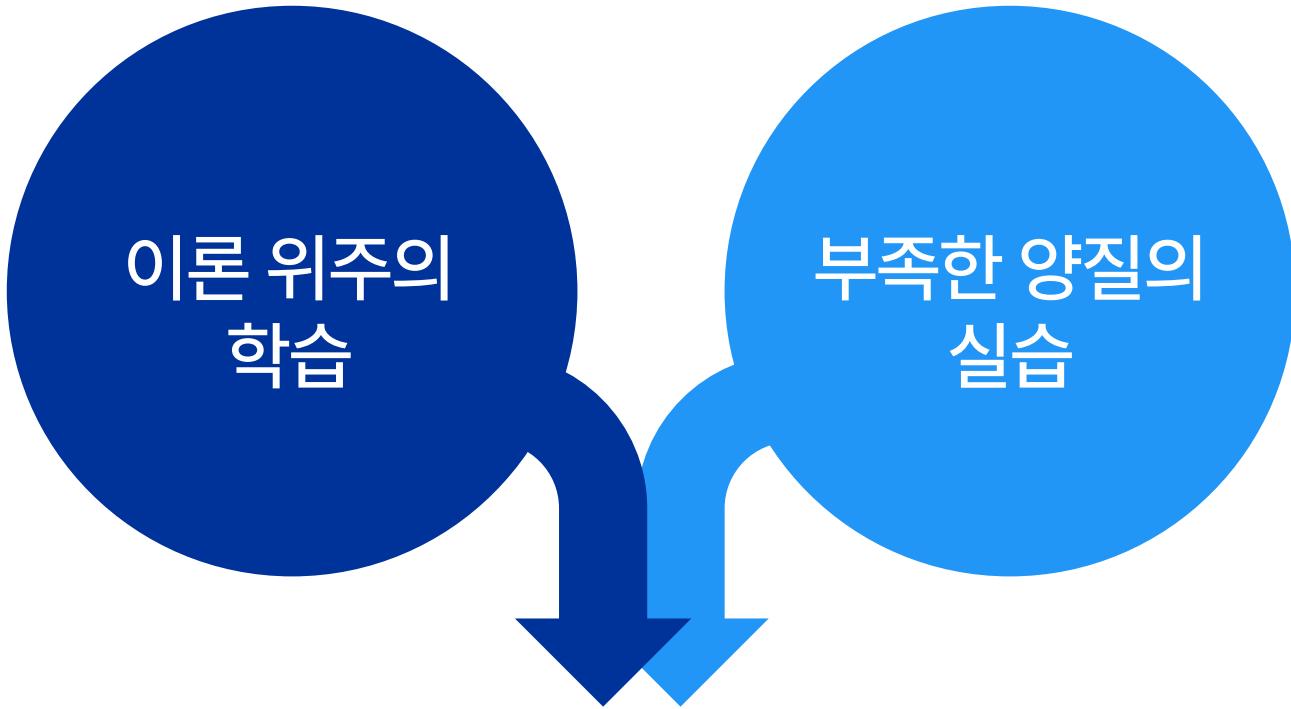
실습 과정에서 겪는 어려움

```

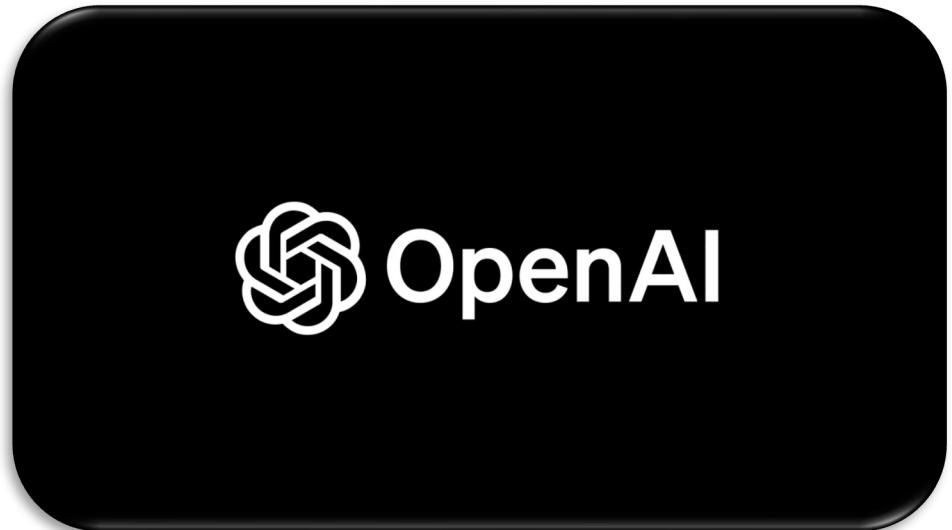
package.json x
package.json > {} scripts
12 "dependencies": {
13     "@radix-ui/react-dialog": "^1.0.0",
14     "@radix-ui/react-avatar": "^1.1.0",
15     "@radix-ui/react-checkbox": "^1.1.1",
16     "@radix-ui/react-dialog": "^1.1.1",
17     "@radix-ui/react-dropdown-menu": "^2.0.6",
18     "@radix-ui/react-hover-card": "^1.1.1",
19     "@radix-ui/react-icons": "^1.3.0",
20     "@radix-ui/react-label": "^2.1.0",
21     "@radix-ui/react-popover": "^1.0.7",
22     "@radix-ui/react-select": "^2.1.1",
23     "@radix-ui/react-slot": "^1.0.2",
24     "@radix-ui/react-tabs": "^1.1.0",
25     "@radix-ui/react-toast": "^1.2.1",
26     "@radix-ui/react-tooltip": "^1.1.2",
27     "qsupabase/supabase-js": "^2.45.1",
28     "@tanstack/react-table": "^8.13.2",
29     "@uidotdev/usehooks": "^2.4.1",
30     "class-variance-authority": "^0.7.0",
31     "clsx": "^2.1.0",
32     "embla-carousel-auto-scroll": "^8.0.0",
33     "embla-carousel-autoplay": "^8.0.0",
34     "embla-carousel-react": "^8.0.0",
35     "framer-motion": "^11.0.8",
36     "jsonwebtoken": "^9.0.2",
37     "lucide-react": "^0.335.0",
38     "next": "^14.2.7",
39     "next-auth": "^5.0.0-beta.20",
40     "path-to-regexp": "^8.0.0",
41     "react": "^18.2.0",
42     "react-dom": "^18.2.0",
43     "react-easy-edit": "^1.17.2",
44     "react-hook-form": "7.53.0",
45     "react-icons": "5.3.0",
46     "react-intersection-observer": "9.8.1",
47     "react-rating-stars-component": "2.2.0",
48     "tailwind-merge": "2.2.1",
49     "tailwindcss-animate": "1.0.7",
50     "vault": "0.9.0",
51     "zod": "3.22.4"
52 },
53 "devDependencies": {
54     "@types/jsonwebtoken": "^9.0.6",
55     "@types/node": "^20",
56     "@types/react": "^18.2.57",
57     "@types/react-dom": "^18.2.19",
58     "autoprefixer": "10.0.1",
59     "eslint": "8",
60 }
61 }

```





문제 해결 능력 저하



01 / 사용자 맞춤 문제 생성

02 / Q&A 형식의 이론 설명

03 / 코드에 대한 피드백 제공

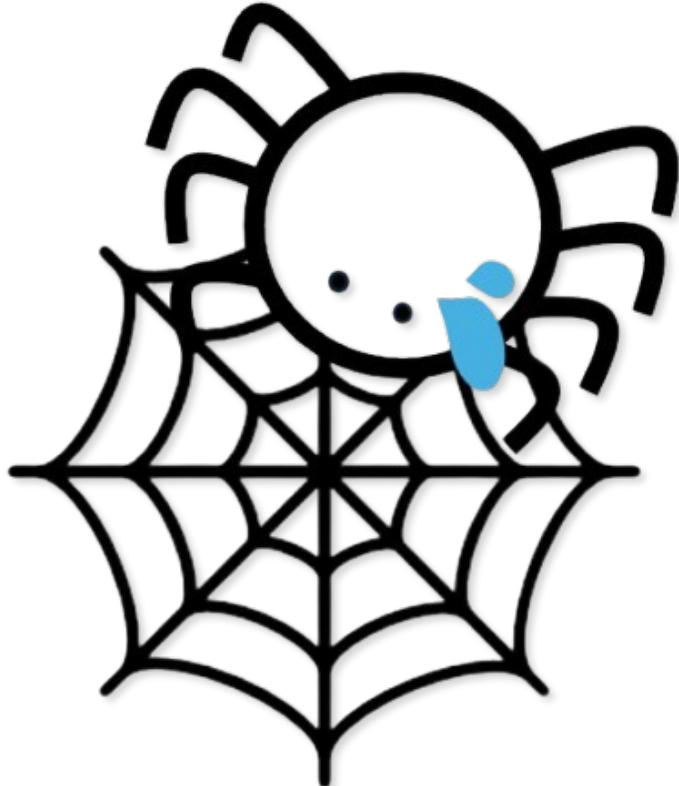
Contents

01 Goals

› 02 Methods

03 Plans

WebShooter



01 개인 맞춤형 학습

02 차별화된 학습 경험

03 Task 수행 능력 강화

1. 개인 맞춤형 학습

전체	출처	단계	분류	추가된 문제	더 보기 ▾
단계별 관리자					
<ul style="list-style-type: none"> jhnhah917 (2023년 9월 11일 - 현재) jh05013 (2019년 6월 24일 - 2023년 9월 10일) 					
단계	제목	설명			
1	입출력과 사칙연산	입력, 출력과 사칙연산을 연습해 봅시다. Hello World!			
2	조건문	if 등의 조건문을 사용해 봅시다.			
3	반복문	for, while 등의 반복문을 사용해 봅시다.			
4	1차원 배열	배열을 사용해 봅시다.			
5	문자열	문자열을 다루는 문제들을 해결해 봅시다.			
6	심화 1	지금까지의 프로그래밍 문법으로 더 어려운 문제들을 풀어봅시다.			
7	2차원 배열	배열 안에 배열이 있다면 어떨까요? 2차원 배열을 만들어 봅시다.			
8	일반 수학 1	수학적 사고력을 길러 봅시다.			
9	약수, 배수와 소수	약수와 배수는 정수론의 시작점이라고 할 수 있습니다.			
10	기하: 직사각형과 삼각형	간단한 도형으로 기하 문제풀이를 시작해 봅시다.			
11	시간 복잡도	프로그램의 정확한 실행 시간을 예측하기는 매우 어렵습니다. 하지만 시간 복잡도를 사용하여 대략적인 예측은 가능합니다.			
12	브루트 포스	가장 간단한 알고리즘인, 모든 경우의 수를 검사하는 브루트 포스 알고리즘을 배워 봅시다.			
13	정렬	배열의 원소를 순서대로 나열하는 알고리즘을 배워 봅시다.			
14	집합과 맵	특정 원소가 속해 있는지 빠르게 찾거나, 각 원소에 대응되는 원소를 빠르게 찾는 자료구조를 배워 봅시다.			
15	약수, 배수와 소수 2	정수론의 세계로 조금 더 들어가 봅시다.			
16	스택, 큐, 데크	스택, 큐, 데크 자료구조를 사용하여 문제를 해결해 봅시다.			

기존 코딩 학습 플랫폼

학습자의 수준이나 관심사에 상관 없이
동일한 문제집이나, 일률적인 학습 과정을 제공

WebShooter 의 차별점

사용자가 지정한 주제와 난이도를 바탕으로
LLM이 사용자 맞춤형 문제를 생성

2. 차별화된 학습 경험

질문 1 30 XP

브라우저의 콘솔에 30 을 출력하는 코드로 알맞은 것을 선택하세요.

- ① console.say(30);
- ② console.log(30);
- ③ console.print(30);
- ④ console.write(30);
- ⑤ console.output(30);

정답 확인

해설 보기 (-20XP)

질문 2 30 XP

Codeit 이라는 문자열을 brand 라는 변수에 할당한 코드로 알맞은 것을 선택하세요.

- ① brand = Codeit;
- ② Codeit = brand;
- ③ 'Codeit' = brand;
- ④ let brand = 'Codeit';
- ⑤ let Codeit = 'brand';

정답 확인

0/2 힌트 사용

힌트 보기 (-1XP)

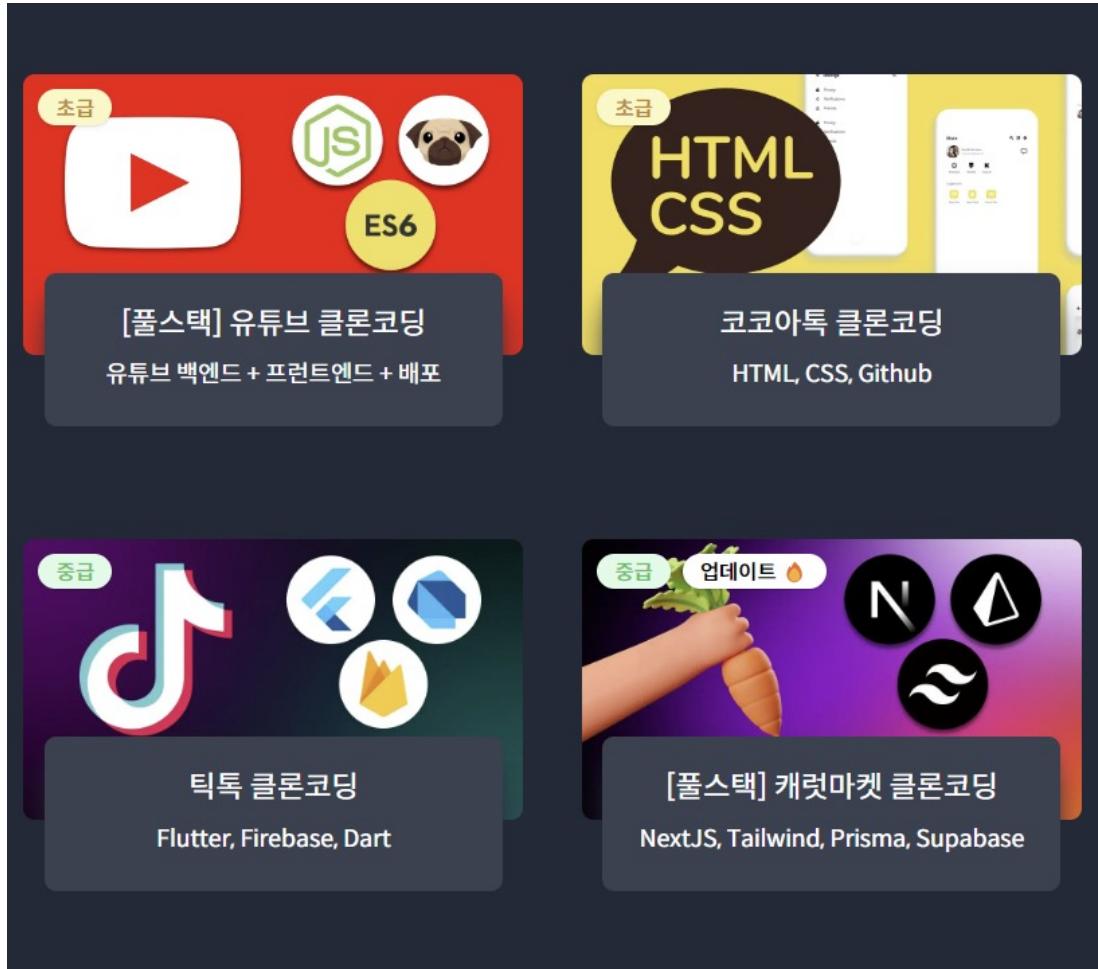
기존 웹 개발 교육 플랫폼

채점 시스템의 부재로 인한 수동 리뷰 대기,
이론 중심의 평가나 단순 퀴즈 형식의 학습 제공

WebShooter 의 차별점

웹 개발 실습 환경에서의 채점 시스템 구축
LLM을 활용한 채점 결과에 대한 상세한 피드백 제공

3. Task 수행 능력 강화



기존 웹 개발 강의 및 실습

프로젝트 전체를 수동적으로 따라 치는 클론 코딩
실무에서 요구되는 코드 분석 및 구현 실습의 부족

WebShooter 의 차별점

특정 컴포넌트 개발 및 서비스 로직 구현과 같은
세부적인 문제 해결 능력을 키울 수 있는 실습 제공

01 간단한 개념 제공 및 QA 기능

LLM을 이용하여
유형별 핵심 키워드에 대한 간단한 설명 생성 & 제공

02 유형, 나이별 개인 맞춤 문제 생성

LLM 프롬프트에 유저의 학습 history 고려한
적절한 유형, 나이도의 키워드를 사용하여 문제 생성 & 제공

03 웹 개발 실습 환경 제공

컴포넌트 렌더링을 확인할 수 있는 Live Preview,
API 요청과 응답을 확인할 수 있는 API 테스트 도구 제공

04 채점, 피드백 기능

LLM이 생성한 테스트 코드 사용,
스크린 샷 유사도 등 채점기준 사용

05 유저의 학습 수준 관리

유형, 나이별 유저의 학습 history 관리

06 코드 공유 및 리뷰 게시판

생성된 문제는 일회성이지만 학습자가 원하는 경우 저장, 공유 가능
게시판에 문제 등록하고 다른 유저들과 교류

02 / METHODS

UI/UX

React > Components와 Prop

문제 풀이 AI 챗봇

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1
Webshooter index #2
Webshooter index #3

App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     <>
8       <Shooter index={1} />
9       <Shooter index={2} />
10      <Shooter index={3} />
11    </>
12  );
13}
```

제출하기

Live Preview

Webshooter index #1
Webshooter index #2
Webshooter index #3

React > Components와 Prop

문제 풀이 AI 챗봇

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

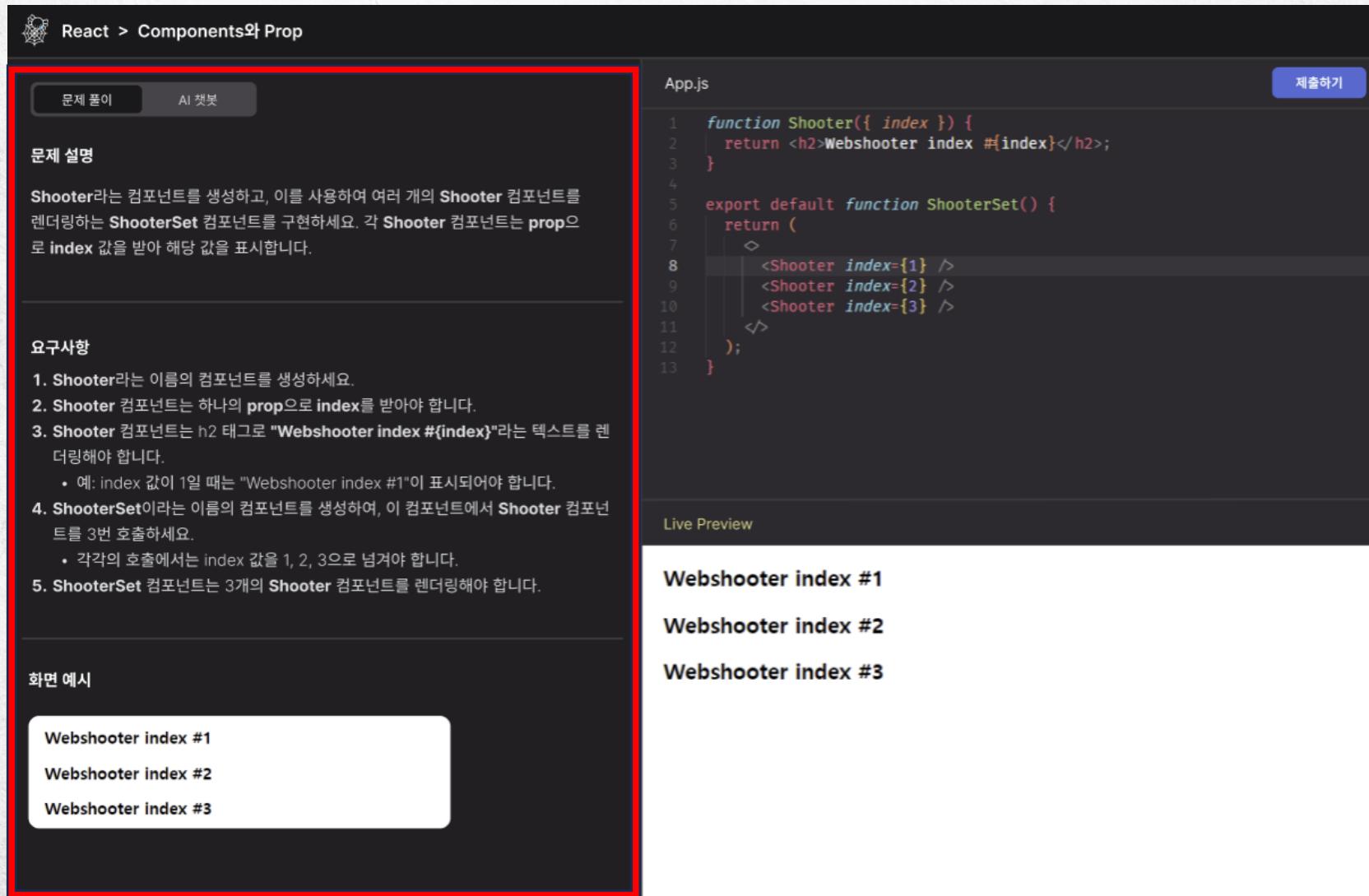
```
App.js
```

```
function Shooter({ index }) {  
  return <h2>Webshooter index #{index}</h2>;  
}  
  
export default function ShooterSet() {  
  return (  
    <>  
      <Shooter index={1} />  
      <Shooter index={2} />  
      <Shooter index={3} />  
    </>  
  );  
}
```

제출하기

Live Preview

Webshooter index #1
Webshooter index #2
Webshooter index #3



문제 풀이 Section

- 실습 문제에 대한 설명
- 구현을 위한 자세한 요구사항
- 모범 정답 화면 예시

02 / METHODS

UI/UX

The screenshot shows a React code editor interface with the following components:

- Header:** React > Components와 Prop
- Toolbars:** 문제 풀이 (highlighted), AI 챗봇
- Code Editor:** A code editor window titled "App.js" containing the following code:

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     <>
8       <Shooter index={1} />
9       <Shooter index={2} />
10      <Shooter index={3} />
11    </>
12  );
13 }
```
- Buttons:** 제출하기 (highlighted)
- Live Preview:** Displays the rendered output of the code:

Webshooter index #1
Webshooter index #2
Webshooter index #3
- Side Panels:** Problem Statement, Requirements, and Examples.

Code Editor

- 스켈레톤 코드 제공
- 사용자가 코드를 작성
- 작성한 코드를 제출
- CodeMirror로 구현

02 / METHODS

UI/UX

The screenshot shows a React development environment. On the left, there's a sidebar with tabs for '문제 풀이' (Problem Solving) and 'AI 챗봇' (AI Chatbot). Below that is a '문제 설명' (Problem Description) section containing instructions for creating a 'Shooter' component and a 'ShooterSet' component. The main area is titled 'App.js' and contains the following code:

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     <>
8       <Shooter index={1} />
9       <Shooter index={2} />
10      <Shooter index={3} />
11    </>
12  );
13}
```

To the right of the code editor is a 'Live Preview' window, which is highlighted with a red border. It displays the rendered output of the code: "Webshooter index #1", "Webshooter index #2", and "Webshooter index #3".

Live Preview

- 작성한 코드에 대한 미리보기
- 핫 리로딩
- react-runner로 구현

02 / METHODS

UI/UX

The screenshot shows a React development interface. On the left, there's a sidebar with tabs for '문제 풀이' and 'AI 챗봇'. Below the tabs is a logo of a spider on a web. A red box highlights the 'AI 챗봇' tab. To its right, the main area displays the code for 'App.js' and its 'Live Preview'.

App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     <>
8       <Shooter index={1} />
9       <Shooter index={2} />
10      <Shooter index={3} />
11    </>
12  );
13}
```

Live Preview

Webshooter index #1
Webshooter index #2
Webshooter index #3

AI 챗봇 Section

- 키워드 기반 추천 질문
- 질의응답 시스템
- RAG 를 이용해 응답의 출처를 사용자에게 제공

02 / METHODS

UI/UX

React > Components와 Prop

문제 풀이 AI 챗봇

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1
Webshooter index #2
Webshooter index #3

App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     <>
8       <Shooter index={1} />
9       <Shooter index={2} />
10      <Shooter index={3} />
11    </>
12  );
13 }
```

제출하기

Live Preview Result

테스트 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)
테스트 1: 통과
테스트 2: 통과
테스트 3: 통과
테스트 4: 통과

채점 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)
테스트 합계: 100/100

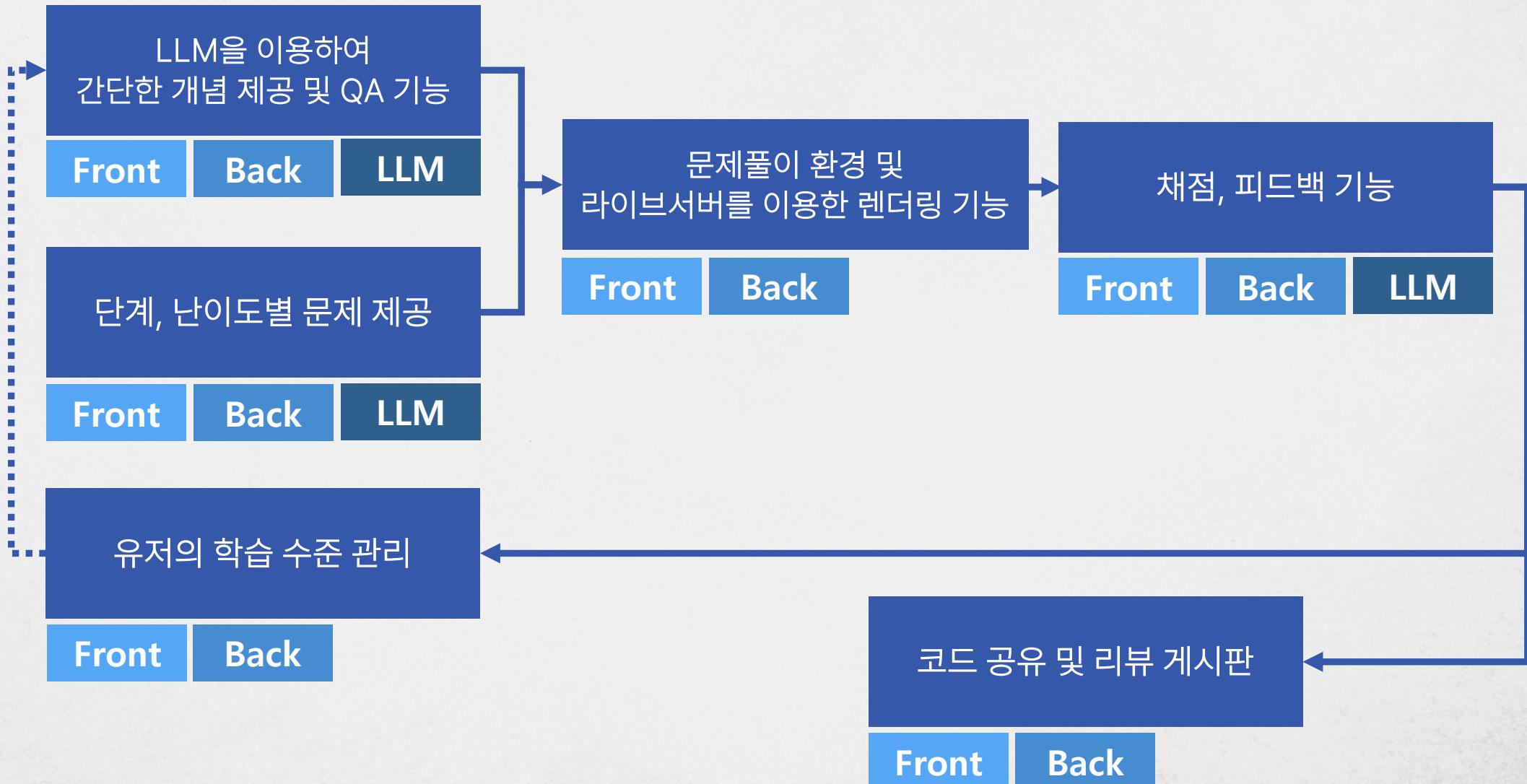
공유하기

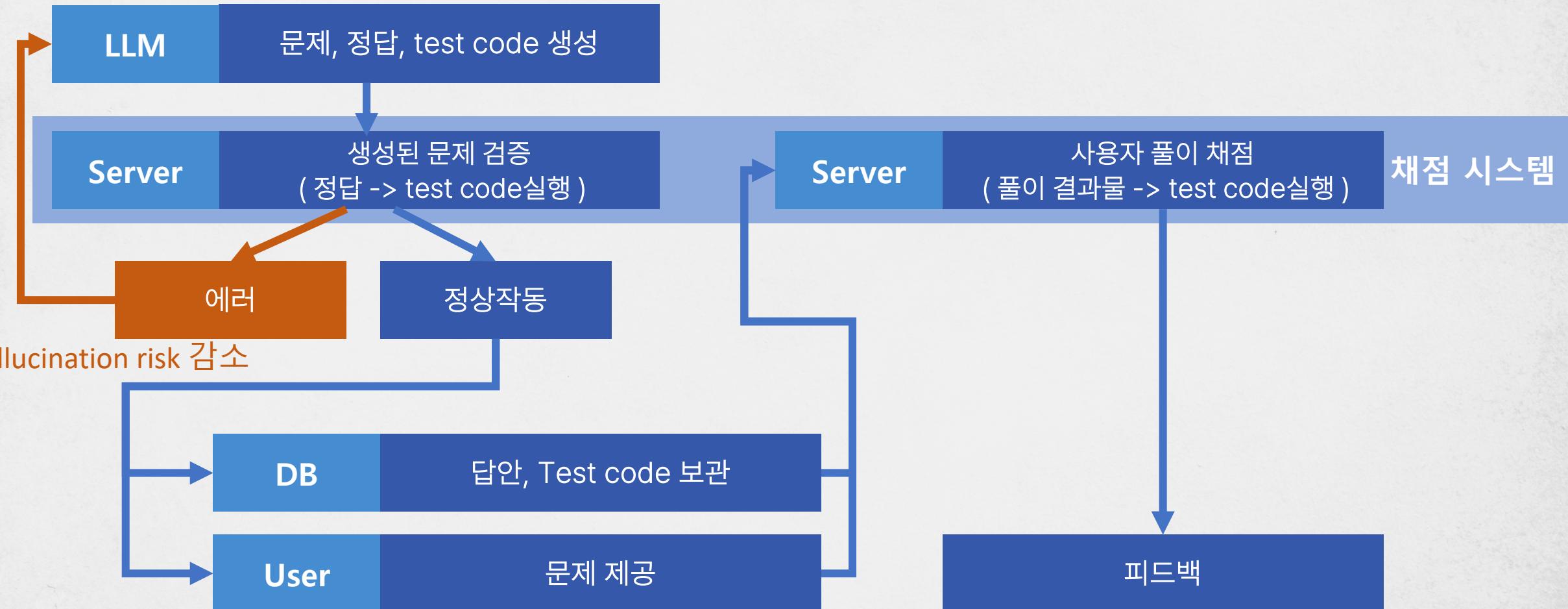
채점 결과

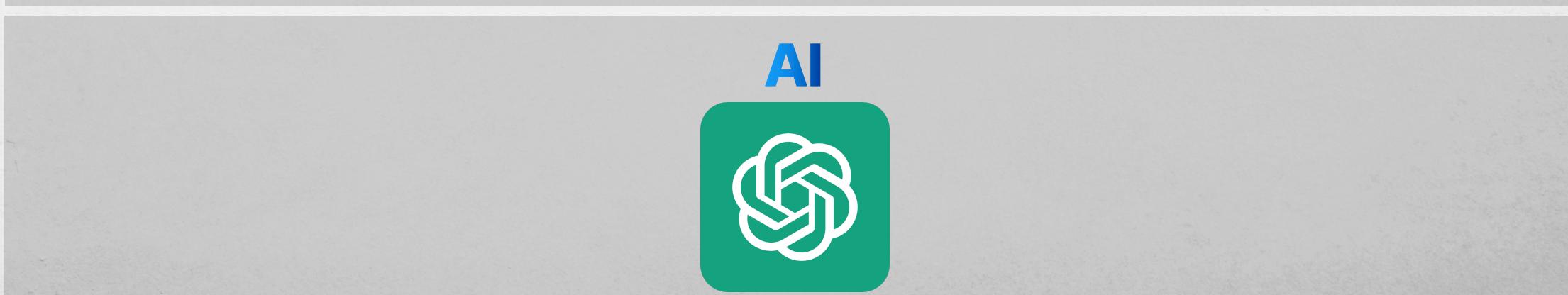
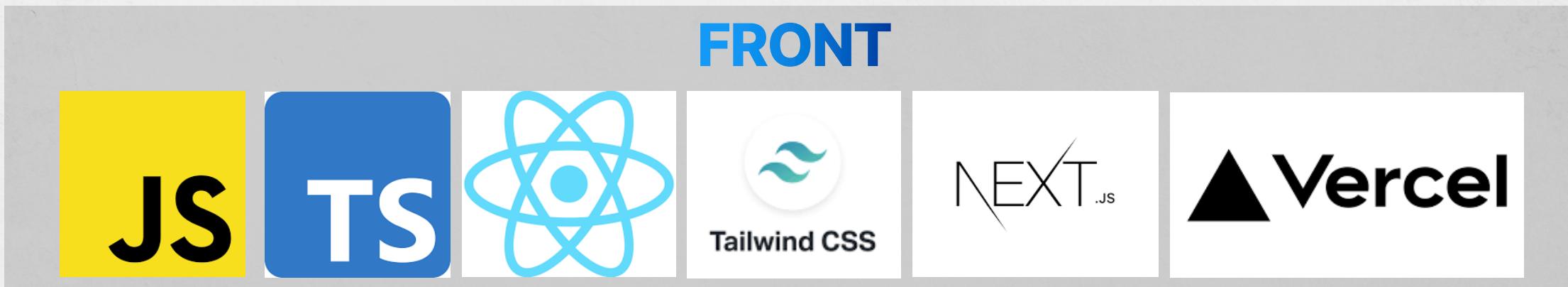
- UI 유사도 분석 결과
- 테스트 코드 통과 여부
- 게시판에 문제 공유

02 / METHODS

ARCHITECTURE







TEAM ROLE

FRONT DESIGN

권서진

FRONT
DEVELOPMENT

권서진, 김현진

BACK (Server, DB)
DEVELOPMENT

이유진, 박경일

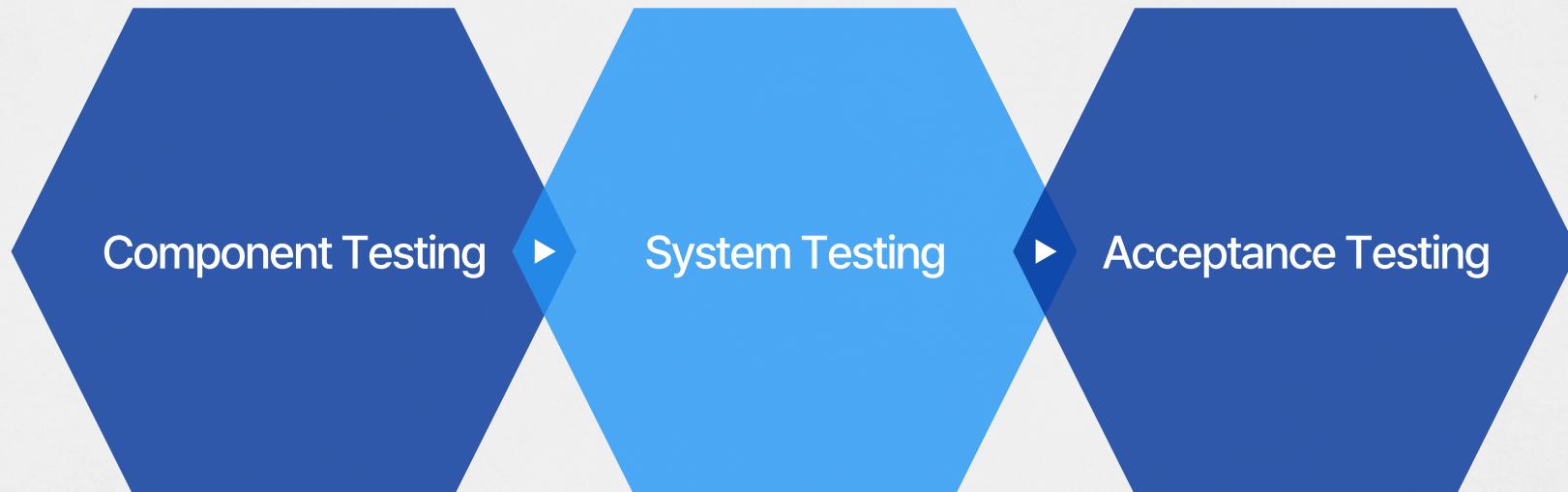
AI (LLM)

박경일, 김문수

OVERALL PLAN

	Week7	Week8	Week9	Week10	Week11	Week12	Week13	Week14
Requirement Specification								
Design Specification								
Requirement Partitioning								
Test case definition								
Preparation for presentation								

TEST PLAN



감사합니다

