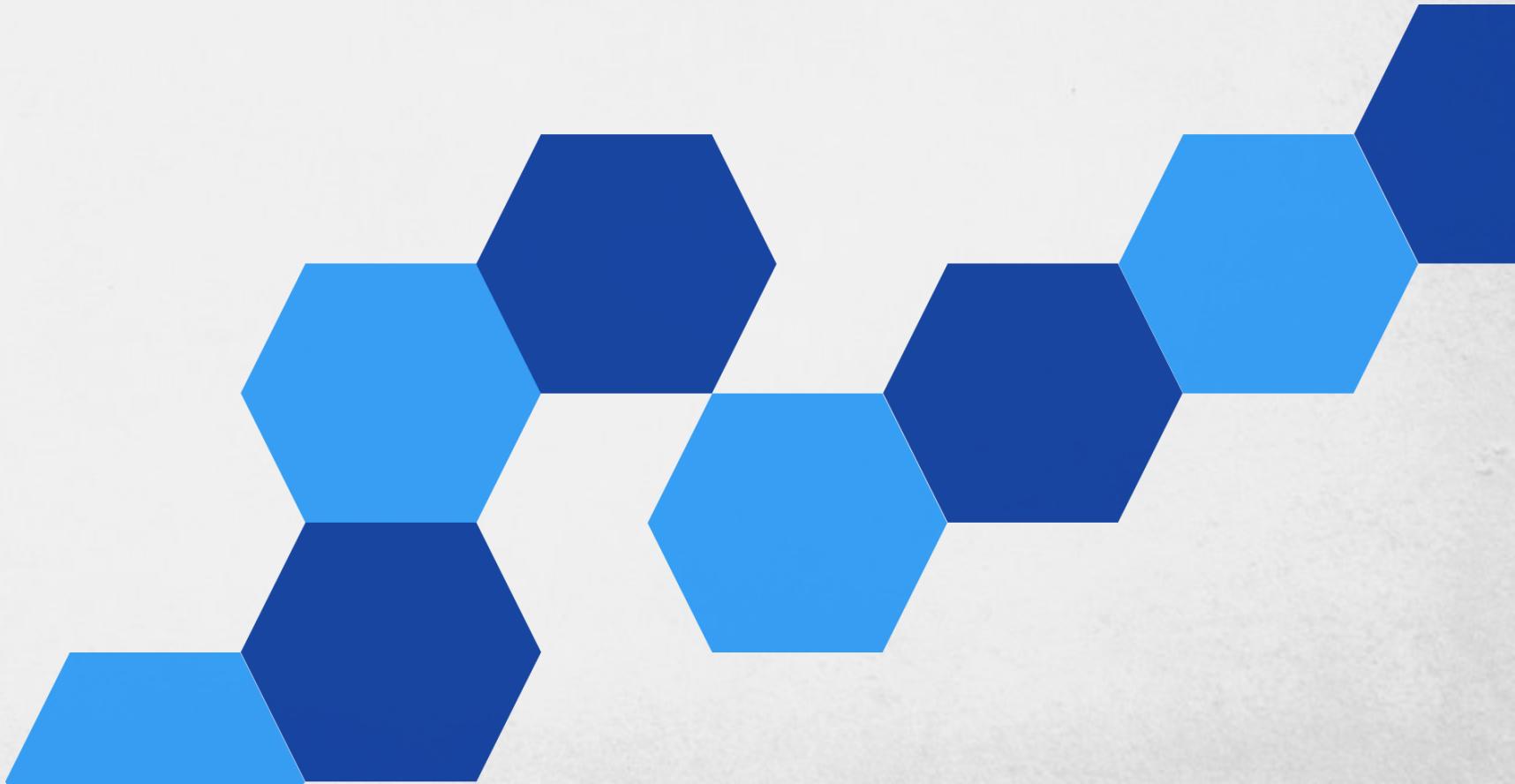


WebShooter

소프트웨어공학개론 1조

권서진, 김문수, 김현진, 박경일, 이유진



Contents

01 Intro

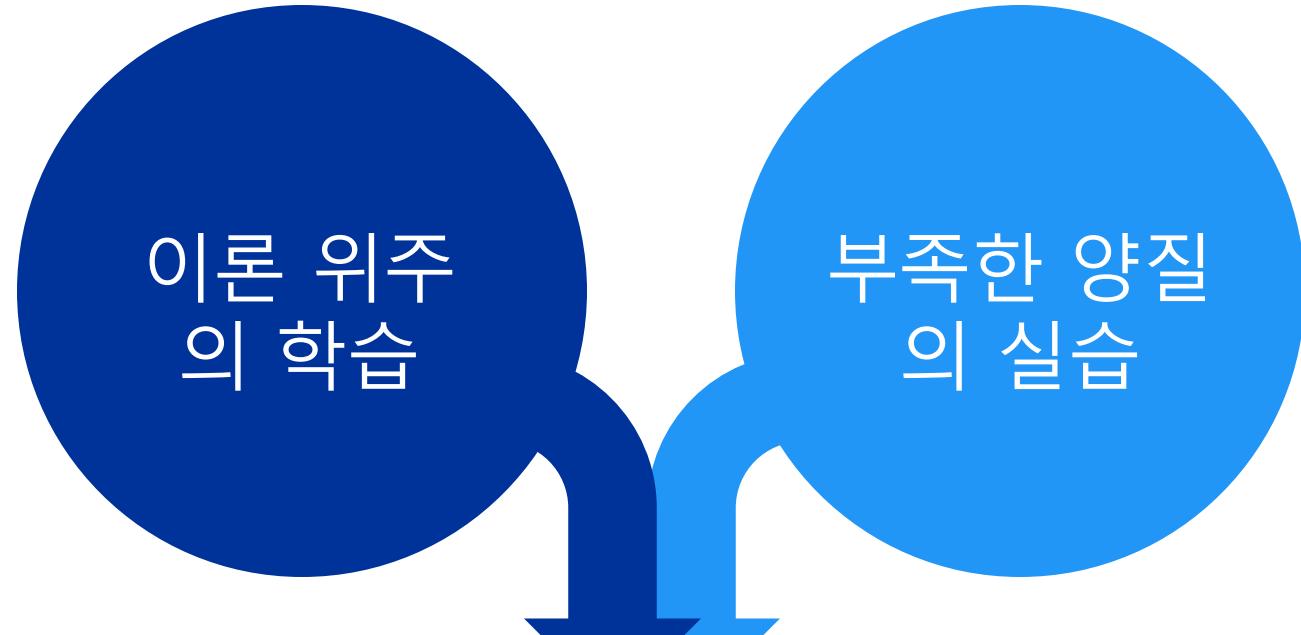


02 Methods



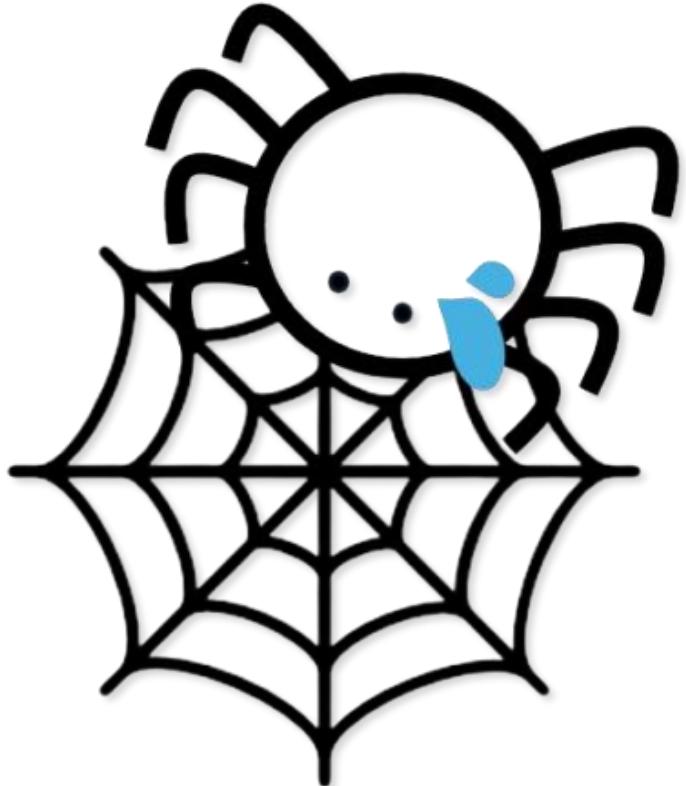
03 Test Design





문제 해결 능력
저하

WebShooter



01 개인 맞춤형 학
습

02 차별화된 학습
경험

03 Task 수행 능력 강
화

01 Home

Home Screen, Login/Signup, Find Password

Problem 02

Filter(Skill/Level)

03 Dashboard

Profile/Learning Progress, Created Problems

Code Editor 04

Problem, Live Preview/API Test Tool, Chatbot, Result, Comment, Line Comment

05 Admin

Problem Management, User Management

01 Home

Home Screen, Login/Signup, Find Password

The screenshot shows the homepage of the WebShooter platform. At the top, there is a dark header bar with the text "WebShooter" on the left and "로그인 회원가입" (Login Member Sign Up) on the right, both in white. Below the header, the word "WebShooter" is prominently displayed in a large, bold, black font. Underneath it, a descriptive paragraph in Korean reads: "WebShooter는 웹 개발을 위한 문제 생성 및 채점 시스템으로 실시간으로 문제를 풀고 피드백을 받을 수 있는 플랫폼입니다. 다양한 웹 개발 이론과 기술을 학습하고, 문제를 해결해보세요!" (WebShooter is a problem generation and grading system for web development. It is a platform where you can solve problems in real-time and receive feedback. Learn various web development theories and technologies, and solve problems!). Below this text is a blue button labeled "문제 생성하기" (Create Problem). The main content area features a grid of six colored boxes representing different levels and categories of problems:

- Level 1**: "Button UI 생성하기" (Create Button UI) - Tags: JSX
- Level 3**: "갤러리 에러 수정" (Gallery Error Fix) - Tags: state, Event Handling
- Level 4**: "Form Validation" - Tags: zod, schema
- Level 2**: "컴포넌트 생성" (Component Creation) - Tags: JSX
- Level 5**: "채팅방 재연결 로직" (Chat Room Reconnection Logic) - Tags: 커스텀 Hooks, Effect
- Level 3**: "갤러리 에러 수정" (Gallery Error Fix) - Tags: state, Event Handling)

A red box highlights the first three columns of these problem cards.

Home

- 로그인, 회원가입
- 전체 실습 문제
- 문제 생성

02 / METHODS

UI / UX

WebShooter

WebShooter

WebShooter는 웹 개발을 위한 문제 생성 및 채점 시스템으로
실시간으로 문제를 풀고 피드백을 받을 수 있는 플랫폼입니다.
다양한 웹 개발 이론과 기술을 학습하고, 문제를 해결해보세요!

문제 생성하기

전체 실습 문제

Level 1

Button UI 생성하기

JSX

Level 3

갤러리 에러 수정

state Event Handling

Level 4

Form Validation

zod schema

Level 2

컴포넌트 생성

JSX

Level 5

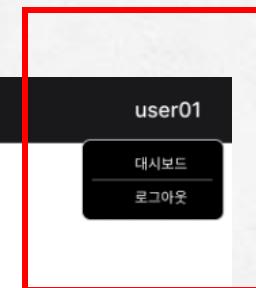
채팅방 재연결 로직

커스텀 Hook Effect

Level 3

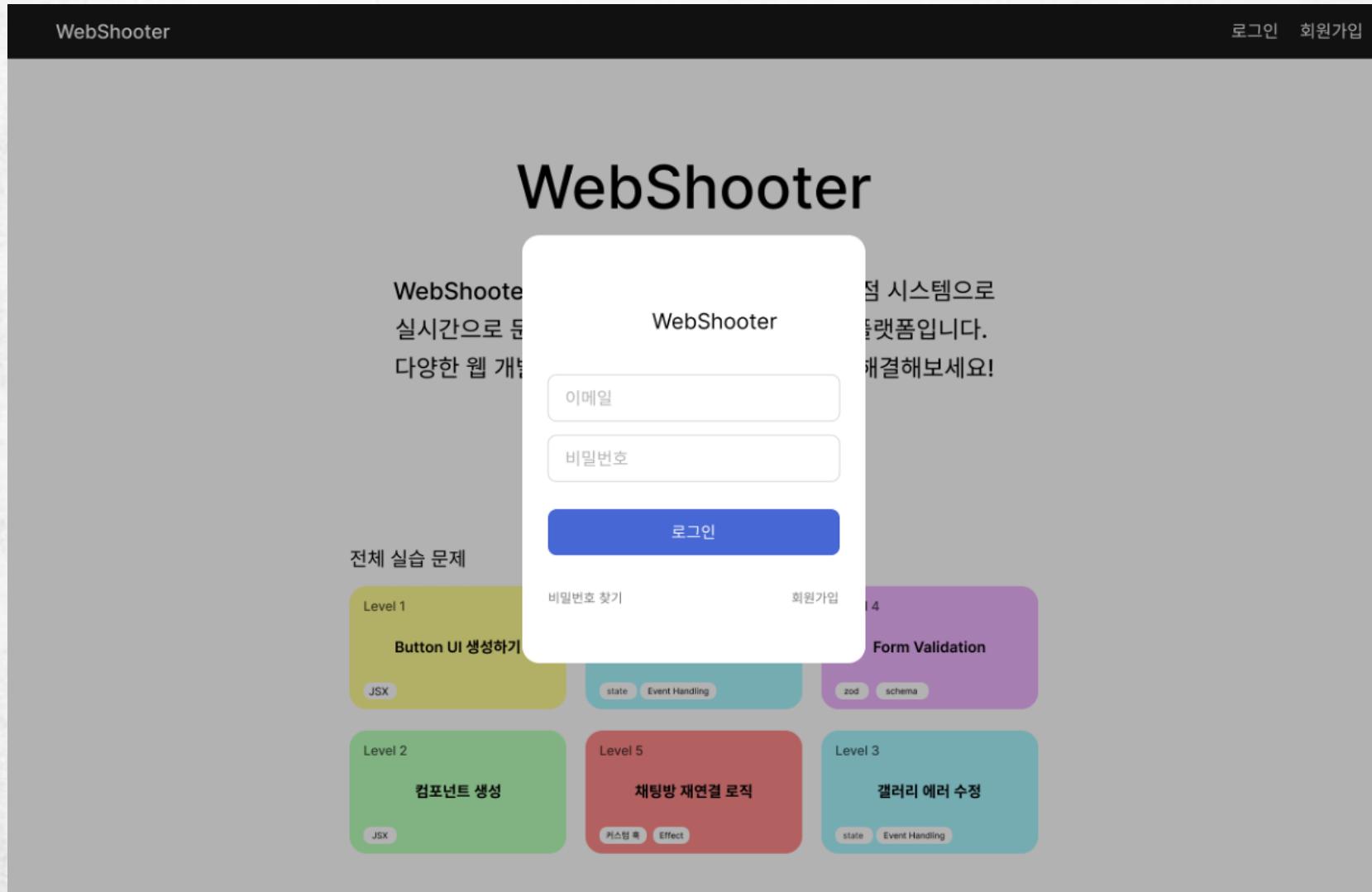
갤러리 에러 수정

state Event Handling



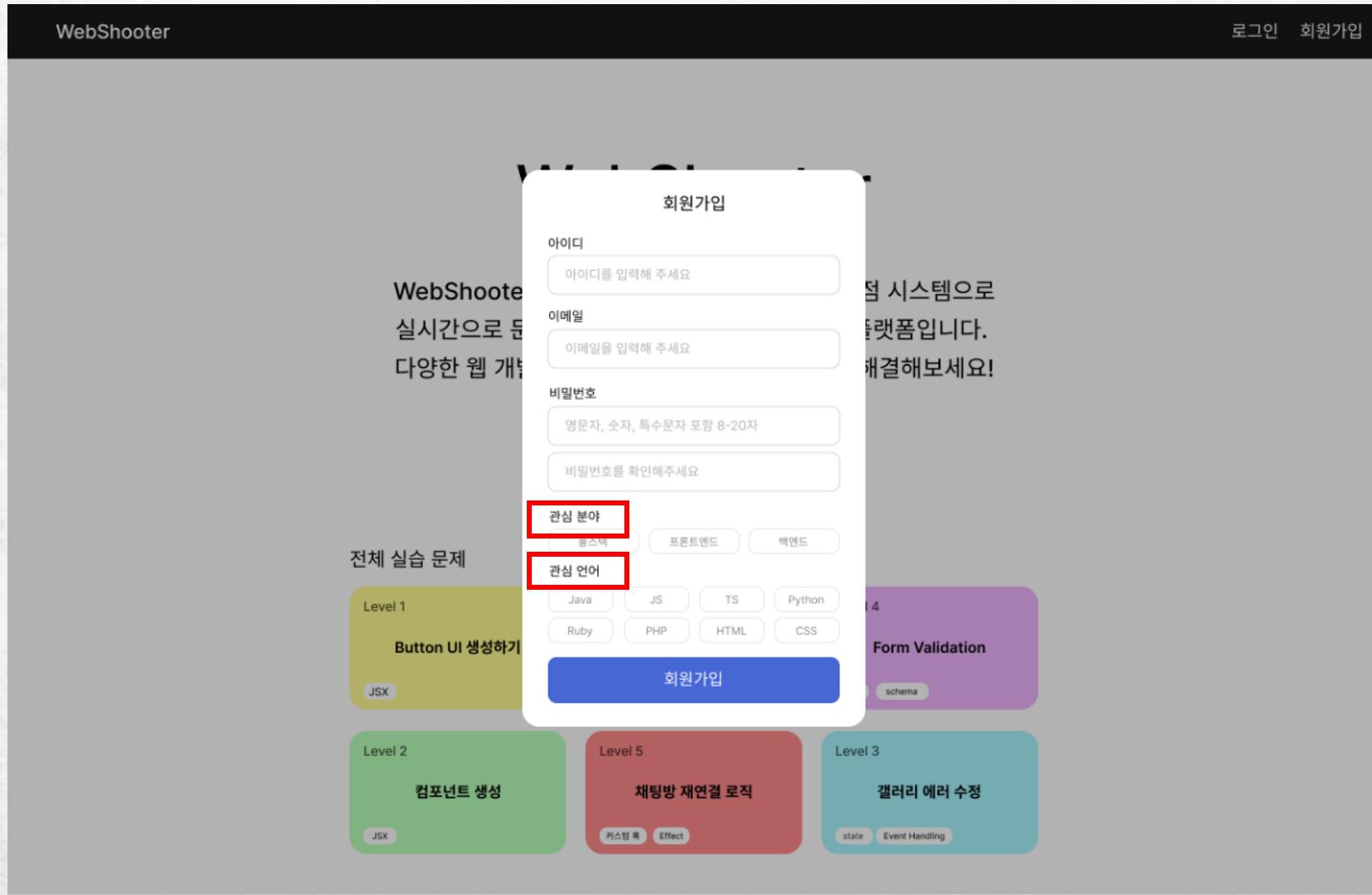
Home

- 로그인 시, username 표시
- 개인의 대시 보드 및 로그아웃



Home : Login

- 로그인 시, 가입한 이메일과 비밀번호를 입력 받음
- 로그인 창에서 비밀번호 찾기, 회원가입 가능



Home : Signup

- 회원가입 창
- 아이디: 플랫폼 내 username
- 이메일, 비밀번호 입력을 받음
- 회원 가입 시 관심 분야, 관심 언어를 입력 받음으로써 관련된 문제를 추천 받을 수 있음



Home

: Find Password

- 비밀번호 분실 시 비밀번호 찾기
 - 이메일로 인증 코드가 발송되고, 해당 인증 코드를 입력함으로써 비밀번호 재설정



Home : Find Password

- 이메일로 발송된 인증 코드
입력 후 새 비밀번호 설정
- 새 비밀번호 확인을 통해 수
정된 비밀번호 검토

Problem 02

Filter(Skill/Level)

The screenshot shows a user interface for a learning platform called WebShooter. At the top, there is a dark header bar with the text "WebShooter" on the left and "로그인 회원가입" (Login/Join) on the right. Below the header, the main content area displays a grid of 32 learning modules, each represented by a colored box with a title, a brief description, and some tags at the bottom.

Filtering Options: A red box highlights the filter section at the top left, which includes a dropdown for "기술 유형" (Skill Type) and a dropdown for "난이도" (Difficulty Level).

Grid Layout:

- Row 1:** Level 1 (Yellow box) - Button UI 생성하기 (JSX); Level 2 (Green box) - 컴포넌트 생성 (JSX); Level 3 (Cyan box) - 갤러리 에러 설정 (state, Event Handling); Level 4 (Purple box) - Form Validation (zod, schema)
- Row 2:** Level 5 (Red box) - 채팅방 재연결 로직 (커스텀 훅, Effect); Level 3 (Cyan box) - 재사용 가능한 컴포넌트 (JSX); Level 3 (Cyan box) - 고급 이벤트 처리 (JSX); Level 3 (Cyan box) - 다크 모드 및 동적 스타일링 (UI, UX)
- Row 3:** Level 4 (Purple box) - 상태 관리와 버튼 연동 (JSX); Level 4 (Purple box) - Custom Hooks (JSX); Level 5 (Red box) - 디자인 시스템 컴포넌트 통합 (JSX); Level 5 (Red box) - 애니메이션과 접근성 향상 (JSX)

A small downward arrow icon is located at the bottom center of the grid area.

Problem - Filter

- 전체 실습 문제에 대한 리스트업
- 실습 문제는 기술 유형과 난이도로 필터링할 수 있어 개인 맞춤형 학습이 가능

WebShooter

로그인 회원가입

전체 32건

기술 유형

- JSX
- State
- Event Handling
- 커스텀 흐
- Effect
- UI
- UX
- zod
- schema
- Prop
- Component
- Controller
- Service
- Domain
- Repository
- Unit Test
- Form
- HTTP

세상의 세간을 도록

세상의 기술을 핵심으로

커스텀 흐 Effect

JSX

Level 3

갤러리 에러 설정

state Event Handling

Level 4

Form Validation

zod schema

Level 3

고급 이벤트 처리

JSX

Level 3

다크 모드 및 동적 스타일링

UI UX

Level 4

상태 관리와 버튼 연동

JSX

Level 4

Custom Hooks

JSX

Level 5

디자인 시스템 컴포넌트 통합

JSX

Level 5

애니메이션과 접근성 향상

JSX

▽

Problem - Filter : Skill Filter

- 기술 유형은 구성요소, 상태 관리, 스키마 등 웹 개발의 기본적인 지식이 되는 분야로 설정
- 프론트엔드/백엔드 관련된 주제 설정

WebShooter

로그인 회원가입

전체 32건

기술 유형 난이도

Level 1 Button UI 생성하기

JSX

Level 1 생성

JSX

Level 3 갤러리 에러 설정

state Event Handling

Level 4 Form Validation

zod schema

Level 5 채팅방 재연결 로직

커스텀 흐 Effect

Level 3 재사용 가능한 컴포넌트

JSX

Level 3 고급 이벤트 처리

JSX

Level 3 다크 모드 및 동적 스타일링

UI UX

Level 4 상태 관리와 버튼 연동

JSX

Level 4 Custom Hooks

JSX

Level 5 디자인 시스템 컴포넌트 통합

JSX

Level 5 애니메이션과 접근성 향상

JSX

Problem - Filter : Level Filter

- Level 은 5단계로 필터링
- 난이도에 따라 분류함으로써 단계적 학습이 가능하도록 함
- Level에 따라 문제의 색상을 다르게 하여 직관성을 높임

The screenshot shows a user interface for a skill filtering system. At the top, there's a black header bar with the text "WebShooter" on the left and "로그인 회원가입" on the right. Below the header, the main content area has a title "전체 8건" (Total 8 items) and two filter buttons: "JSX" and "난이도 ▾" (Difficulty ▾). The content is organized into a grid of 8 cards:

- Level 1**: "Button UI 생성하기" (JSX)
- Level 2**: "컴포넌트 생성" (JSX)
- Level 3**: "재사용 가능한 컴포넌트" (JSX)
- Level 3**: "고급 이벤트 처리" (JSX)
- Level 4**: "상태 관리와 버튼 연동" (JSX)
- Level 4**: "Custom Hooks" (JSX)
- Level 5**: "디자인 시스템 컴포넌트 통합" (JSX)
- Level 5**: "애니메이션과 접근성 향상" (JSX)

A red box highlights the "JSX" button in the first card's row.

Problem :Skill Filter Applied

- 기술 유형 필터링이 적용된 모습
- 문제 컴포넌트 왼쪽 하단에 해당 기술유형 표기

02 / METHODS

UI / UX

The screenshot shows a list of six Level 3 components on the WebShooter platform:

- 갤러리 애러 설정 (Level 3) - Tags: state, Event Handling
- 재사용 가능한 컴포넌트 (Level 3) - Tags: JSX
- 고급 이벤트 처리 (Level 3) - Tags: JSX
- 라우팅 처리 (Level 3) - Tags: UX
- 전역 상태 관리 (Level 3) - Tags: Redux, Zustand
- 다크 모드 및 동적 스타일링 (Level 3) - Tags: UI, UX

A red box highlights the "Level 3" label above the first component, indicating it is the applied filter.

Problem :Level Filter Applied

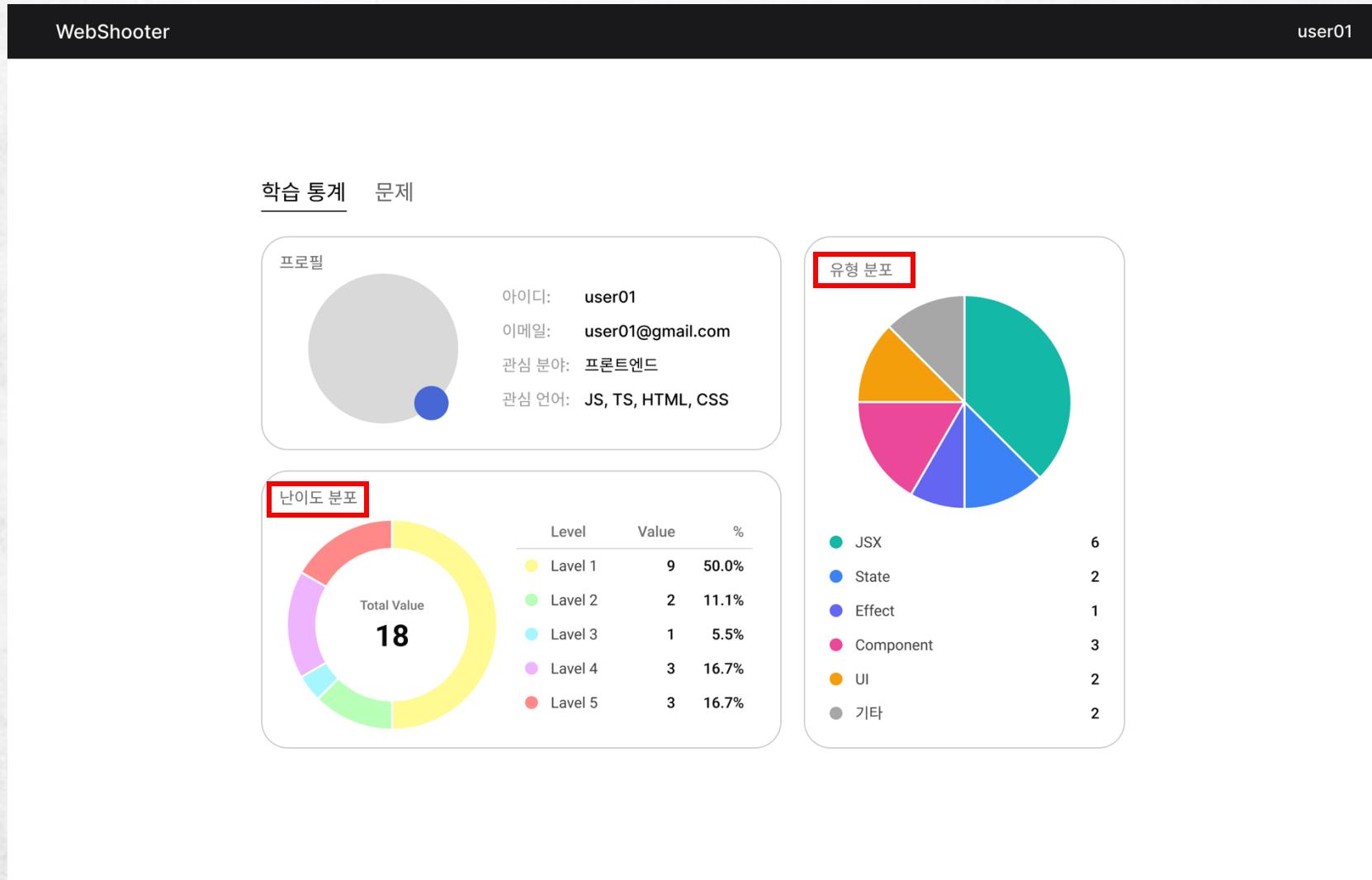
- 난이도 필터링이 적용된 모습
- 문제 컴포넌트 왼쪽 상단에 해당 난이도 표기

03 Dashboard

Profile/Learning Progress, Created Problems

02 / METHODS

UI / UX

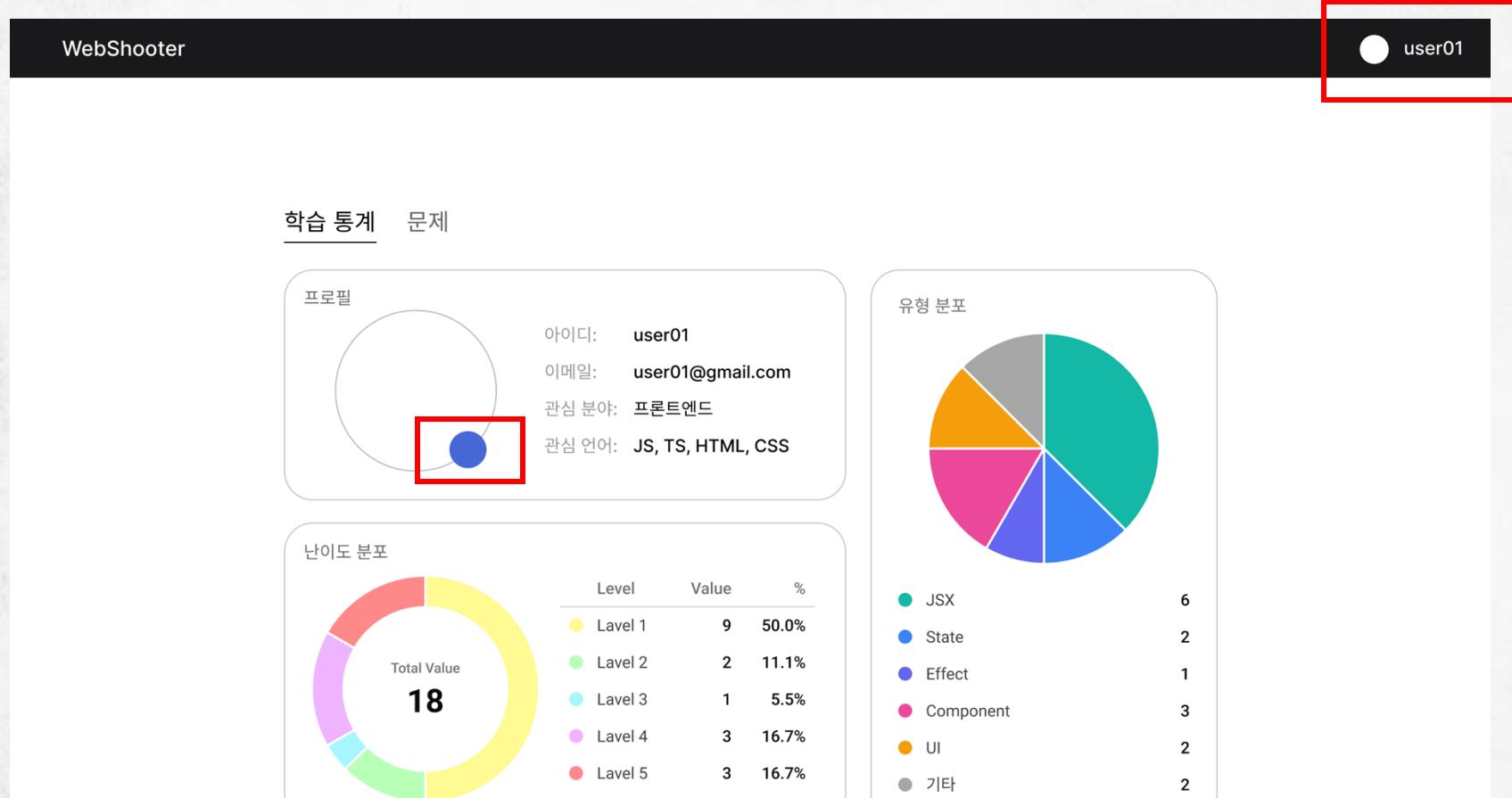


Dashboard : Learning Progress

- 로그인 후, 개인 대시보드에서 개인의 학습 진척도 확인 가능
- 회원가입 시 설정한 분야, 언어 확인
- 학습한 문제, 학습한 기술 유형 등을 통계로 나타냄

02 / METHODS

UI / UX



Dashboard : Profile Update

- 프로필 이미지 업데이트
- 상단 username 옆에 표시

02 / METHODS

UI / UX

The screenshot shows a dark-themed dashboard interface. At the top, there is a black header bar with the text "WebShooter" on the left and "user01" on the right. Below the header, the main content area has a white background. On the left side of the main area, there is a navigation menu with the text "학습 통계" and "문제" (Problem) under it. The "문제" item is highlighted with a red rectangular box. In the center of the main area, there is a section titled "user01 가 생성한 문제 5건" (5 problems generated by user01). This section contains a table with the following data:

난이도	제목	정답률
Level 1	Button UI 생성하기	100.0%
Level 3	다크 모드 및 동적 스타일링	80.0%
Level 4	상태 관리와 버튼 연동	66.7%
Level 5	디자인 시스템 컴포넌트 통합	83.3%
Level 5	애니메이션과 접근성 향애니메이션과 접근성 향상	37.5%

Dashboard : Created Problems

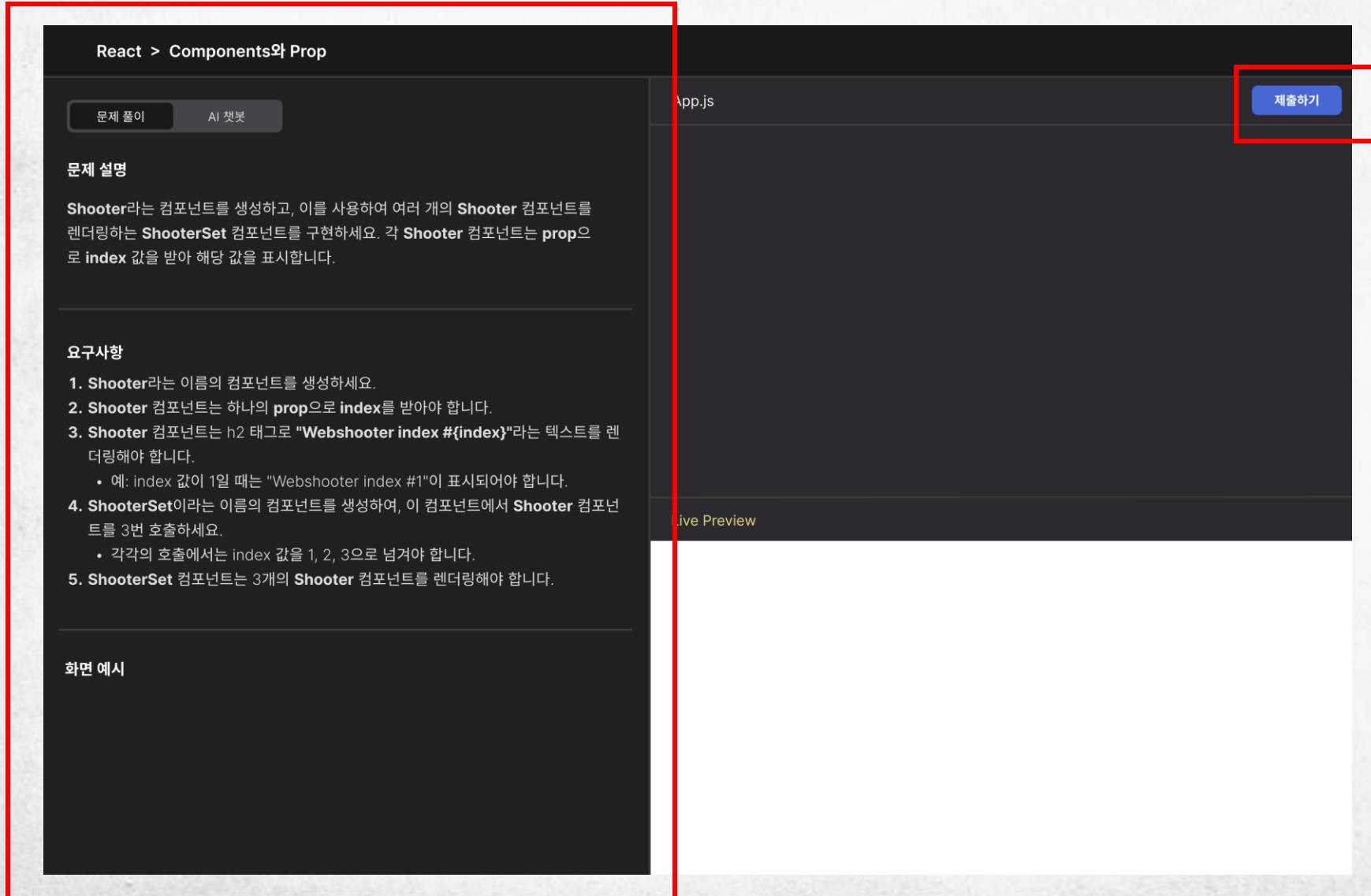
- 개인이 LLM을 통해 생성한 문제 확인

Code Editor 04

Problem, Live Preview/API Test Tool, Chatbot,
Result, Comment, Line Comment

02 / METHODS

UI / UX



Code Editor : Problem

- 문제 화면
- 왼쪽 창에서 문제 제목, 문제 설명, 요구사항, 화면 예시 확인
- 오른쪽 창에서 코드 작성 및 제출

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

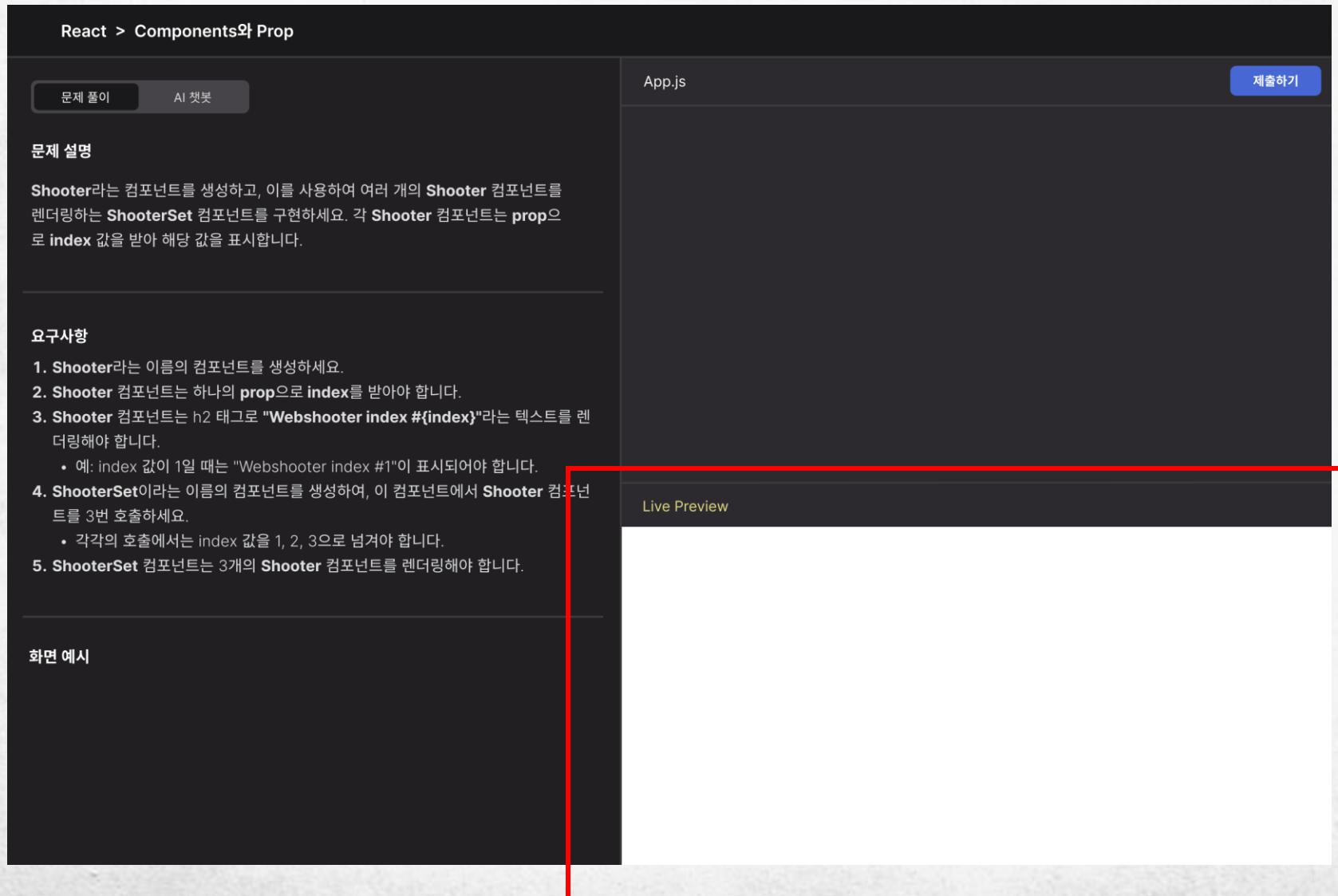
1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.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

화면 예시

App.js

제출하기

Live Preview



Code Editor : Live Preview

- 프론트엔드 실습 문제
- Live Preview를 통해 코드 작성을 실시간으로 반영

02 / METHODS

UI / UX

React > Components와 Prop

문제 풀이 AI 챗봇

문제 설명

/problem 엔드포인트에 GET 요청을 처리하는 API를 구현하세요. 이 API는 다양한 문제 목록을 담고 있는 배열을 JSON 형식으로 반환해야 합니다. 반환되는 문제들은 각 문제의 고유 ID, 제목, 난이도를 포함해야 합니다.

요구사항

- 엔드포인트: /problem
 - HTTP 메서드: GET
 - 문제 목록 배열을 반환합니다. 각 문제는 고유 ID, 제목, 난이도를 포함해야 합니다.
- 문제 데이터:
 - 각 문제는 다음의 속성들을 가져야 합니다:
 - id: 고유한 문제 ID (정수형)
 - title: 문제 제목 (문자열)
 - difficulty: 문제 난이도 (문자열, 예: "쉬움", "중간", "어려움")

응답 예시

```
[  
  {  
    "id": 1,  
    "title": "간단한 API 구현",  
    "difficulty": "쉬움",  
  },  
  {  
    "id": 2,  
    "title": "데이터베이스 연결",  
    "difficulty": "중간",  
  },  
  {  
    "id": 3,  
  }]
```

Problem.java ProblemController.java

제출하기

API Test Tool

GET {{ baseUrl }}/problem Send

Query Body Headers Auth Vars Response Headers

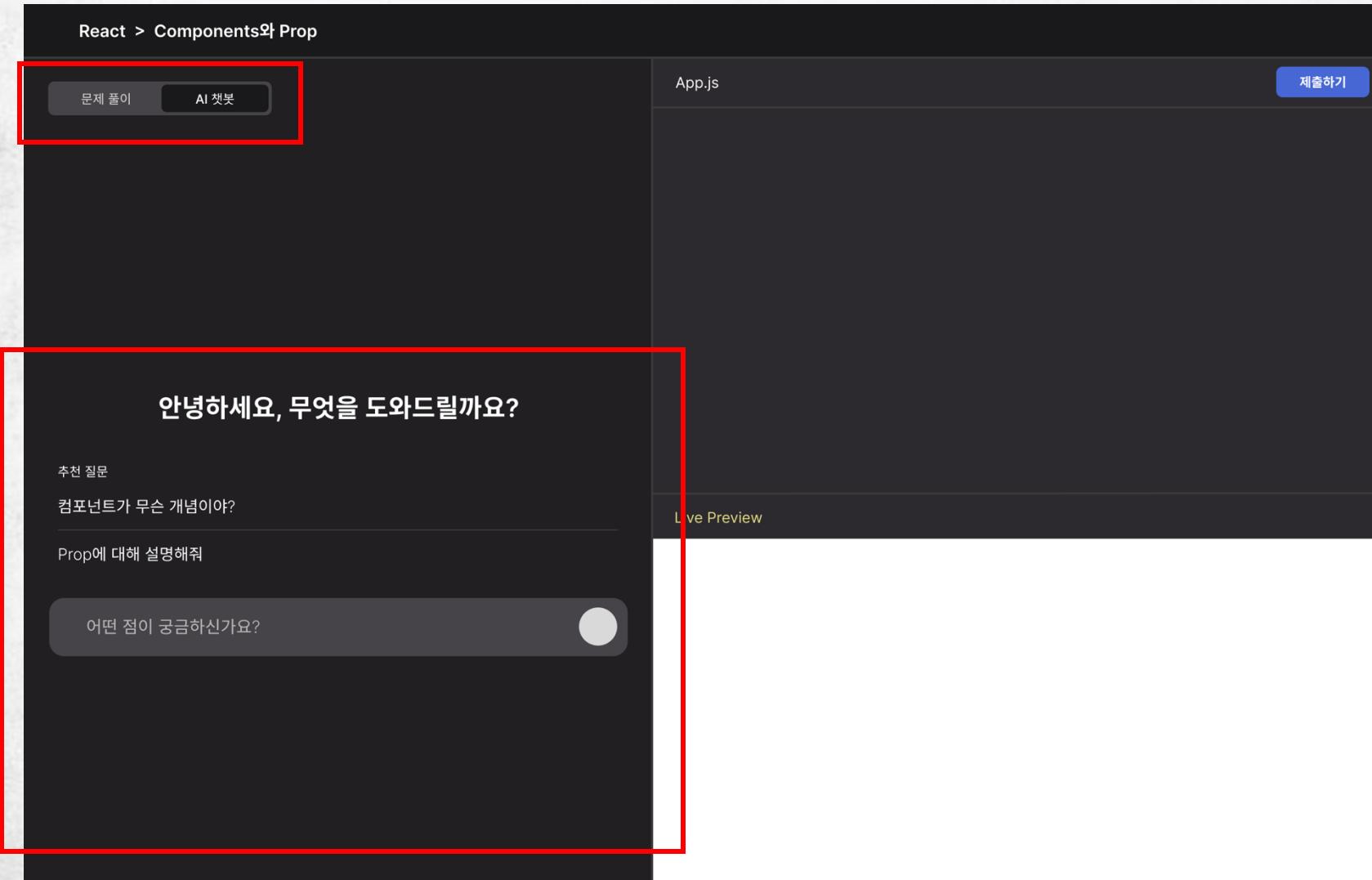
No Body

Code Editor : API Test Tool

- 백엔드 실습 문제
- API Test Tool을 통해, API 개발 및 테스트 실습

02 / METHODS

UI / UX



Code Editor : Chatbot

- LLM 기반 Chatbot과 실시간 질의응답 가능

02 / METHODS

UI / UX

React > Components와 Prop

문제 풀이 AI 챗봇

Prop에 대해 설명해줘

React에서 **Prop** (Properties)은 컴포넌트 간에 데이터를 전달하는 방법입니다. React 컴포넌트는 재사용 가능한 UI 요소로 구성되며, 각 컴포넌트는 다른 컴포넌트로 데이터를 전달할 수 있습니다. 이때 사용되는 것이 props입니다.

Prop의 주요 개념

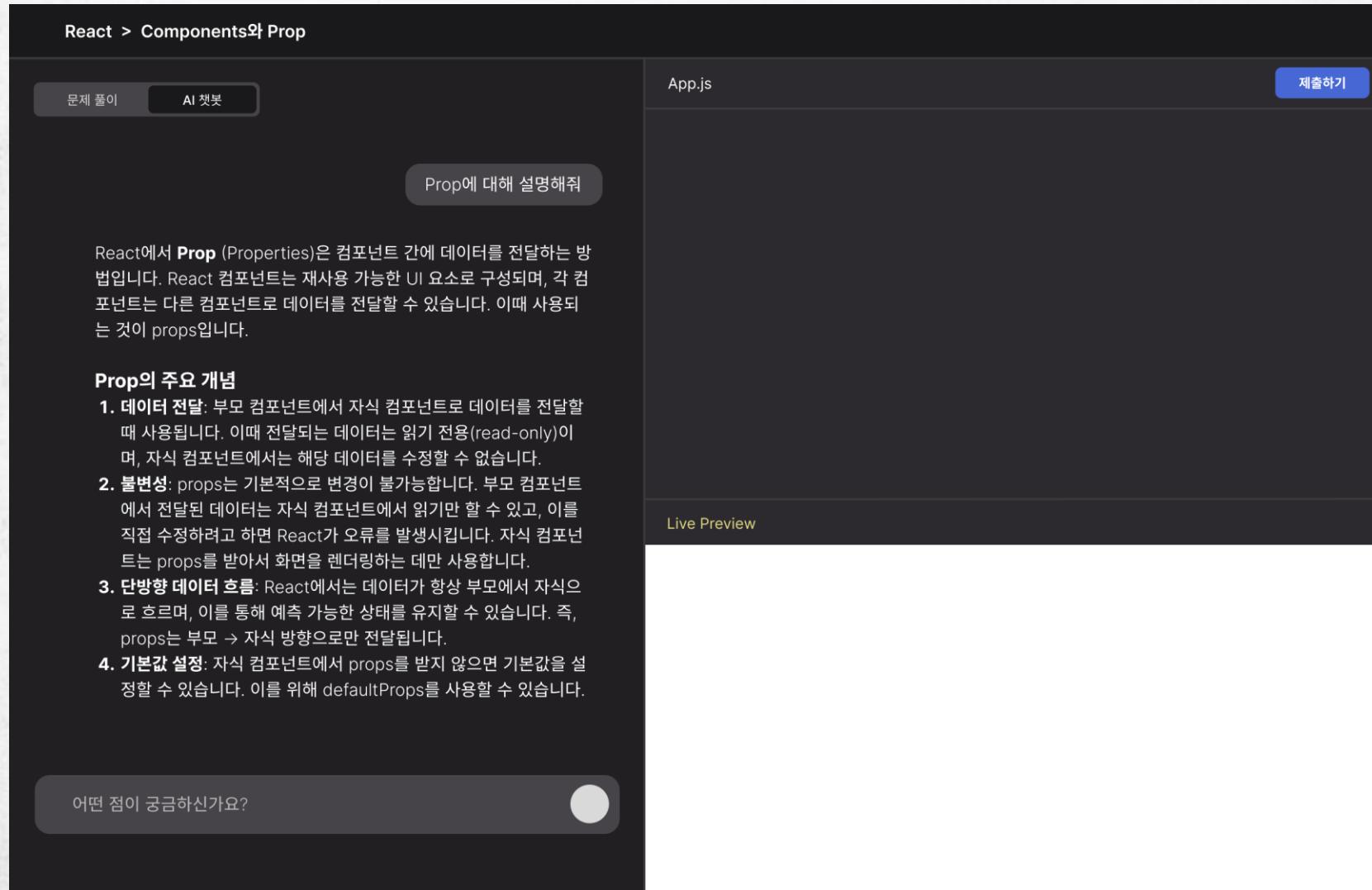
- 데이터 전달:** 부모 컴포넌트에서 자식 컴포넌트로 데이터를 전달할 때 사용됩니다. 이때 전달되는 데이터는 읽기 전용(read-only)이며, 자식 컴포넌트에서는 해당 데이터를 수정할 수 없습니다.
- 불변성:** props는 기본적으로 변경이 불가능합니다. 부모 컴포넌트에서 전달된 데이터는 자식 컴포넌트에서 읽기만 할 수 있고, 이를 직접 수정하려고 하면 React가 오류를 발생시킵니다. 자식 컴포넌트는 props를 받아서 화면을 렌더링하는 데만 사용합니다.
- 단방향 데이터 흐름:** React에서는 데이터가 항상 부모에서 자식으로 흐르며, 이를 통해 예측 가능한 상태를 유지할 수 있습니다. 즉, props는 부모 → 자식 방향으로만 전달됩니다.
- 기본값 설정:** 자식 컴포넌트에서 props를 받지 않으면 기본값을 설정할 수 있습니다. 이를 위해 defaultProps를 사용할 수 있습니다.

어떤 점이 궁금하신가요?

App.js

제출하기

Live Preview



Code Editor : Chatbot

- 챗봇과 질의응답을 통해 관련된 개념에 대한 질의 응답 가능

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

App.js

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

문제 풀이 AI 챗봇

문제 설명

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

요구사항

1. Shooter라는 이름의 컴포넌트를 생성하세요.
2. Shooter 컴포넌트는 하나의 prop으로 index를 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. ShooterSet이라는 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

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

Code Editor : Result

- 제출 결과를 채점 서버로 전송하여 채점
- 제출 코드의 렌더링과 정답 코드의 렌더링 결과를 비교해 유사도를 분석하여 채점

02 / METHODS

UI / UX

React > Components와 Prop

문제 풀이 AI 챗봇

제출하기

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 더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter**를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

문제를 공유하시겠습니까?

문제를 공유하면 다른 사용자들이 여러분의 제출 내역을 볼 수 있게 되고, 그에 대한 피드백을 주고받을 수 있습니다.

Level 3

Components와 Prop

prop Component

취소 확인

준: 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

The screenshot shows a dark-themed code editor interface. At the top, there are tabs for '문제 풀이' and 'AI 챗봇'. On the right, there's a blue button labeled '제출하기'. Below these, the title 'React > Components와 Prop' is displayed. The main area contains a '문제 설명' section with instructions about creating a 'Shooter' component and a 'ShooterSet' component. To the left, there's a '요구사항' section with a numbered list of requirements. On the right, there's a '화면 예시' section showing a modal dialog with a red border. The dialog has a question '문제를 공유하시겠습니까?' (Do you want to share your problem?), a descriptive text about sharing, and a 'Level 3' badge. Below the dialog, there's a snippet of code with 'Components와 Prop' and 'prop Component' buttons, along with '취소' (Cancel) and '확인' (Confirm) buttons. At the bottom, there's a '준: 95%' (Score: 95%) indicator and a list of four test cases all marked as '통과' (Passed). Further down, there's a '채점 결과' (Scoring result) section with 'UI 유사도 분석: 99.9%' and '테스트 합계: 100/100'. A blue '공유하기' (Share) button is located at the bottom right of the modal.

Code Editor : Share

- 제출한 문제 중 정답인 문제는 다른 사용자들과 공유 가능
- 공유를 하게되면 다른 사용자들은 자유롭게 문제 풀이 및 코멘트를 남길 수 있음

02 / METHODS

UI / UX

React > Components와 Prop

문제 풀이 AI 챗봇 게시판

App.js

제출 ID: #324 | 아이디: user13 | 점수: 100 / 100 | 제출 시작: 2024-12-07 22:03

user01 2024-12-08 11:23
함수 선언문 대신에 함수 표현식을 쓰는 건 어떨까요?

user03 2024-12-07 23:48
index 배열을 생성하고 map 함수를 사용하면 반복된 코드를 줄일 수
있을 것 같아요!

user05 2024-12-07 22:32
저랑 풀이가 완전히 동일하네요!

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

댓글을 작성해주세요. 불쾌감을 주는 욕설과 악플은 관리자에 의해 삭제될 수 있습니다.

댓글 등록

A screenshot of a React application interface. At the top, there's a navigation bar with tabs: '문제 풀이', 'AI 챗봇', and '게시판'. The '게시판' tab is active. Below the navigation is a dark sidebar containing user comments and a summary box. A red box highlights the summary box. The summary box contains information about a submission: ID #324, User user13, Score 100/100, and Submission Start 2024-12-07 22:03. It also lists three user comments. The main content area shows a 'Live Preview' and 'Result' tab, with the 'Result' tab selected. Under 'Result', there's a '테스트 결과' (Test Results) section showing UI similarity analysis and test results. At the bottom, there's a '채점 결과' (Scoring Results) section with a UI similarity analysis and a total score of 100/100. A note at the bottom says '댓글을 작성해주세요. 불쾌감을 주는 욕설과 악플은 관리자에 의해 삭제될 수 있습니다.' (Please write a comment. Offensive language and spam will be deleted by the administrator.) and a '댓글 등록' (Comment Register) button.

Code Editor : Comment

- 공유된 문제는 공유 게시판
에 게시되고, 다른 사용자들은 해당 문제에 대해 코멘트
를 작성

02 / METHODS

UI / UX

React > Components와 Prop

문제 풀이 AI 챗봇 게시판

문제 설명

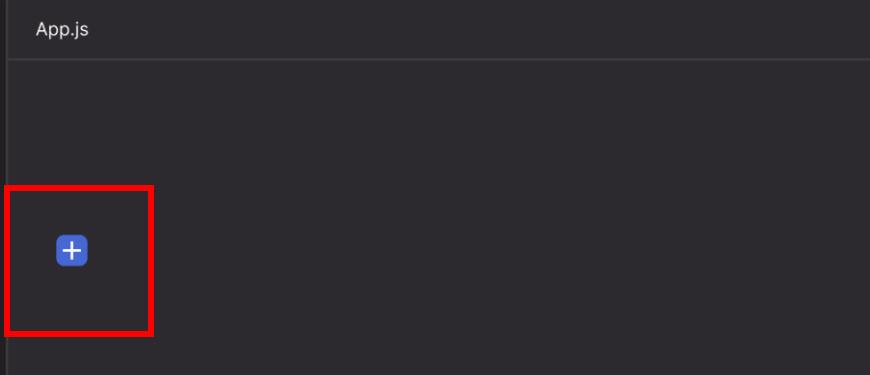
Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

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



```
<ShooterSet>
```

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

Code Editor : Line Comment

- Line Comment를 통해 문제가 아닌 코드에 대한 피드백 가능
- 코드의 해당되는 라인에 +를 누름으로써 Comment 입력

02 / METHODS

UI / UX

React > Components와 Prop

문제 풀이 AI 챗봇 게시판

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 #
더링해야 합니다."입니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1
더링해야 합니다."
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하세요, 이 컴포넌트는 3번 호출하세요.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

App.js

8번 line에 피드백을 추가

피드백을 작성해주세요. 불쾌감을 주는 욕설과 악플은 관리자에 의해 삭제될 수 있습니다.

취소 확인

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

Code Editor : Line Comment

- 모달창이 뜨면서 해당 Line에 피드백을 추가할 수 있음

02 / METHODS

UI / UX

React > Components와 Prop

문제 풀이 AI 챗봇 게시판

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

user03 2024-12-07 23:48
index 배열을 생성하고 map 함수를 사용하면 반복된 코드를 줄일 수 있을 것 같아요!

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

Code Editor : Line Comment

- 해당 Line에 생성된 Comment를 확인할 수 있음

05 Admin

Problem Management, User Management

02 / METHODS

UI / UX

WebShooter Management

admin

문제 사용자

전체 32건

ID	난이도	제목	정답률
#1	Level 1	Button UI 생성하기	13
#2	Level 2	컴포넌트 생성	5
#3	Level 3	갤러리 에러 수정	3
#4	Level 4	Form Validation	2
#5	Level 5	채팅방 재연결 로직	4
#6	Level 3	재사용 가능한 컴포넌트	2
#7	Level 3	고급 이벤트 처리	0
#8	Level 3	다크 모드 및 동적 스타일링	0
#9	Level 4	상태 관리와 버튼 연동	0
#10	Level 4	Custom Hooks	0

1 2 3 4 5

Admin

- Problem Management

- Admin은 전체 문제에 대한
관리 권한을 가짐

02 / METHODS

UI / UX

WebShooter Management

admin

문제 사용자

전체 32건

ID	난이도	제목	정답률
<input type="checkbox"/> #1	Level 1	Button UI 생성하기	13
<input type="checkbox"/> #2	Level 2	문제 삭제 확인	5
<input type="checkbox"/> #3	Level 3	선택한 문제 • 고급 이벤트 처리 • 다크 모드 및 동적 스타일링	3
<input type="checkbox"/> #4	Level 4	문제를 정말 삭제하시겠습니까?	2
<input type="checkbox"/> #5	Level 5	삭제된 데이터는 복구할 수 없습니다.	4
<input type="checkbox"/> #6	Level 3	<input type="button" value="취소"/> <input type="button" value="확인"/>	2
<input checked="" type="checkbox"/> #7	Level 3	고급 이벤트 처리	0
<input checked="" type="checkbox"/> #8	Level 3	다크 모드 및 동적 스타일링	0
<input type="checkbox"/> #9	Level 4	상태 관리와 버튼 연동	0
<input type="checkbox"/> #10	Level 4	Custom Hooks	0

삭제

취소

1 2 3 4 5

Admin

- Problem Management
: Delete Problem

- 사용자가 생성한 문제를 선택하여 삭제

02 / METHODS

UI / UX

The screenshot shows a dark-themed React development interface. At the top, a blue header bar contains the text "React > Components와 Prop". On the right side of the header, the word "admin" is displayed. Below the header, the title "App.js" is centered above a large, mostly empty dark gray workspace. In the bottom left corner of the workspace, the text "Live Preview" is visible. To the left of the workspace, there are three vertical sections with white borders:

- 문제 설명**: Describes the task: creating a **Shooter** component that generates a **ShooterSet** component containing multiple **Shooter** components. Each **Shooter** component receives an **index** prop and displays the text "Webshooter index #{{index}}".
- 요구사항**: A numbered list of requirements:
 1. Create a **Shooter** component.
 2. The **Shooter** component must receive a **prop** named **index**.
 3. The **Shooter** component must render an **h2** tag with the text "Webshooter index #{{index}}".
 4. Create a **ShooterSet** component that generates three **Shooter** components with **index** values of 1, 2, and 3.
 5. The **ShooterSet** component must render three **Shooter** components.
- 화면 예시**: This section is currently empty.

Admin

- Problem Management
- : Problem Detail
- 전체 문제 목록에서 문제를 클릭 시 문제의 상세 정보를 확인할 수 있음

WebShooter Management

문제 사용자 admin

전체 43건

	ID	아이디	권한	생성한 문제 수	생성한 댓글 수
<input type="checkbox"/>	#1	admin	관리자	13	0
<input type="checkbox"/>	#2	user01	일반 사용자	5	4
<input type="checkbox"/>	#3	user02	일반 사용자	3	5
<input type="checkbox"/>	#4	user03	일반 사용자	2	3
<input type="checkbox"/>	#5	user04	일반 사용자	4	2
<input type="checkbox"/>	#6	user05	일반 사용자	2	1
<input type="checkbox"/>	#7	user06	일반 사용자	0	0
<input type="checkbox"/>	#8	user07	일반 사용자	0	0
<input type="checkbox"/>	#9	user08	일반 사용자	0	0
<input type="checkbox"/>	#10	user09	일반 사용자	0	0

삭제

1 2 3 4 5

Admin

– User Management

- 사용자 목록 조회
- 사용자가 작성한 댓글 및 코드 리뷰 조회 및 삭제

02 / METHODS

UI / UX

WebShooter Management

admin

문제 사용자

전체 43건

ID	아이디	권한	생성한 문제 수	생성한 댓글 수
<input type="checkbox"/> #1	admin	관리자	13	0
<input type="checkbox"/> #2	user01	사용자 삭제 확인		
<input type="checkbox"/> #3	user02	선택한 사용자 user08과 user09를 정말 삭제하시겠습니까?		
<input type="checkbox"/> #4	user03	삭제된 데이터는 복구할 수 없습니다.		
<input type="checkbox"/> #5	user04			
<input type="checkbox"/> #6	user05	<input type="button" value="취소"/> <input type="button" value="확인"/>		
<input type="checkbox"/> #7	user06	일반 사용자	0	0
<input type="checkbox"/> #8	user07	일반 사용자	0	0
<input checked="" type="checkbox"/> #9	user08	일반 사용자	0	0
<input checked="" type="checkbox"/> #10	user09	일반 사용자	0	0

The screenshot shows a user management interface titled 'WebShooter Management' with a blue header bar. The user 'admin' is logged in. Below the header, there are two tabs: '문제' (Problems) and '사용자' (Users), with '사용자' being the active tab. A sub-header indicates there are 43 users listed. The main area displays a table with columns: ID, 아이디 (Id), 권한 (Role), 생성한 문제 수 (Created Problem Count), and 생성한 댓글 수 (Created Comment Count). The table lists 10 users, numbered #1 to #10. User #1 is the administrator ('admin'). Users #2 and #3 are regular users ('user01' and 'user02'). A modal dialog box is overlaid on the table, centered over users #8 and #9. The dialog has a white background and a thin gray border. It contains the text '선택한 사용자 user08과 user09를 정말 삭제하시겠습니까?' (Do you really want to delete the selected users, user08 and user09?) and '삭제된 데이터는 복구할 수 없습니다.' (Deleted data cannot be recovered). At the bottom of the dialog are two buttons: '취소' (Cancel) on the left and a larger '확인' (Confirm) button on the right, which is highlighted with a blue background.

Admin

– User Management : Delete User

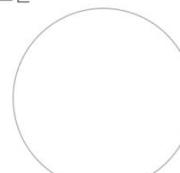
- User에 대한 삭제 권한

WebShooter Management

admin

문제 사용자

프로필



아이디: user01
이메일: user01@gmail.com
관심 분야: 프론트엔드
관심 언어: JS, TS, HTML, CSS

작성한 댓글 / 피드백 3건

댓글 ID	문제 ID	내용	종류
<input type="checkbox"/> #37	#15	함수 선언문 대신에 함수 표현식을 쓰는 건 어떨까요?	댓글
<input type="checkbox"/> #43	#18	현재 코드에서 useState를 사용하고 있는데. 해당 값이...	피드백
<input type="checkbox"/> #47	#18	이 코드는 없어도 똑같이 작동할 것 같아요!	피드백

삭제

1

Admin

– User Management : User Detail

- User의 정보에 대한 상세 정보 확인
- User 가 작성한 댓글, 피드백 확인

02 / METHODS

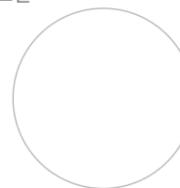
UI / UX

WebShooter Management

admin

문제 사용자

프로필



아이디: user01
이메일: user01@gmail.com
관심 분야: 프론트엔드
관심 언어: JS, TS, HTML, CSS

작성한 댓글 / 피드백 3건

삭제

댓글 ID	문제 ID	내용	종류
<input type="checkbox"/> #37	#15	함수 선언문 대신에 함수 표현식을 쓰는 건 어떨까요?	댓글
<input type="checkbox"/> #43	#18	현재 코드에서 useState를 사용하고 있는데. 해당 값이 컴포넌트의 렌더링과 관련이 없다면 useRef를 사용하는 것 이 더 적합합니다! useRef는 값이 변경되어도 컴포넌트를 재렌더링하지 않기 때문에, 렌더링에 영향을 주지 않으면서 도 값을 유지할 수 있습니다!	피드백
<input type="checkbox"/> #47	#18	이 코드는 없어도 똑같이 작동할 것 같아요!	피드백

1

Admin

– User Management : Comment Detail

- Comment 클릭 시 내용 확인

02 / METHODS

UI / UX

WebShooter Management

admin

문제 사용자

프로필

아이디: user01
이메일: user01@gmail.com
관심 분야: 프론트엔드

댓글 삭제 확인

선택한 피드백#47를 정말 삭제하시겠습니까?
삭제된 데이터는 복구할 수 없습니다.

취소 확인

삭제

작성한 댓글 / 피드백 3건

댓글 ID	문제 ID	내용	종류
<input type="checkbox"/> #37	#15	함수 선언문 대신에 함수 표현식을 쓰는 건 어떨까요?	댓글
<input type="checkbox"/> #43	#18	현재 코드에서 useState를 사용하고 있는데. 해당 값이...	피드백
<input checked="" type="checkbox"/> #47	#18	이 코드는 없어도 똑같이 작동할 것 같아요!	피드백

1

댓글 ID	문제 ID	내용	종류
<input type="checkbox"/> #37	#15	함수 선언문 대신에 함수 표현식을 쓰는 건 어떨까요?	댓글
<input type="checkbox"/> #43	#18	현재 코드에서 useState를 사용하고 있는데. 해당 값이...	피드백
<input checked="" type="checkbox"/> #47	#18	이 코드는 없어도 똑같이 작동할 것 같아요!	피드백

Admin
– User Management
: Delete Comment

- 피드백에 대한 삭제 권한

01 End-to-End Testing

Performance Testing 02

03 Compatibility Testing

Security Testing 04

05 Failover and
Recovery Testing

User Testing 06

01 End-to-End Testing

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_signup():
    # Selenium을 활용한 UI 자동화
    click_element("#signup_button")
    wait_until_element_visible("#signup_email_field", timeout=5s)

    # Partition Testing : ex) 비밀번호 길이와 이메일 형식
    input_text("#signup_id_field", "moonsu99")
    input_text("#signup_email_field", "moonsu99@example.com") # 정상 이메일
    input_text("#signup_password_field", "Moonsu123!") # 정상 비밀번호
    input_text("#signup_password_confirm_field", "Moonsu123!")

    select_checkbox("#interest_fullstack")
    select_checkbox("#interest_frontend")
    select_checkbox("#language_java")
    select_checkbox("#language_js")
    select_checkbox("#language_python")

    click_element("#final_signup_button")
    wait_until_element_visible("#login_email_field", timeout=5s)
    verify_current_url_contains("/login")

    # Postman으로 비정상 입력값 테스트 (이메일 형식 오류)
    api_response = postman_call("POST", "/api/signup", {"email": "user@com", "password": "InvalidPass123"})
    verify(api_response.status_code == 400)
    verify("유효한 이메일" in api_response.body)

    # Chrome DevTools로 세션 쿠키/스토리지 확인
    open_chrome_devtools()
    check_network_logs_for_cookie("session_id")
    verify_cookie_value("session_id", not_empty=True)

    # Manual Test: 세션 유지, 동시 로그인 확인
    # Verification Point: 회원가입 처리시간 5초 이내
```

test_signup

회원가입 기능의 정상 동작 확인

Verification Point

1. 회원가입 및 로그인 처리 시간이 5초 이내에 완료
2. 잘못된 입력값에 대해 오류 메시지가 명확히 표시

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_password_reset():
    # Selenium UI 비밀번호 재설정 접근
    click_element("#login_button")
    click_element("#find_password_button")
    input_text("#find_password_email_field", "moonsu99@example.com")
    click_element("#auth_request_button")

    # Postman으로 비밀번호 재설정 요청 API 확인
    email_send_res = postman_call("POST", "/api/reset_password", {"email": "moonsu99@example.com"})
    verify(email_send_res.status_code == 200)
    verify(mock_email_server_check("moonsu99@example.com"))

    mock_received_code = "123456"
    input_text("#auth_code_input_field", mock_received_code)
    click_element("#auth_code_confirm_button")

    wait_until_element_visible("#new_password_field", timeout=5s)
    input_text("#new_password_field", "NewMoonsu123!")
    input_text("#new_password_confirm_field", "NewMoonsu123!")
    click_element("#update_password_button")
    wait_until_element_visible("#login_email_field", timeout=5s)

    # Verification Point: 비번 재설정 이메일 및 링크 작동 확인
```

test_password_reset

비밀번호 재설정 기능의 정상 동작 확인

Verification Point

1. 비밀번호 재설정 요청 시 이메일이 정상 전송되고 링크가 작동
2. 잘못된 입력값에 대해 오류 메시지가 명확히 표시

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_login():
    click_element("#login_button")
    input_text("#login_email_field", "moonsu99@example.com")
    input_text("#login_password_field", "NewMoonsu123!")
    click_element("#login_submit_button")
    wait_until_element_visible("#user_avatar", timeout=5s)
    verify_element_text_contains("#user_id_display", "moonsu99")
    # Verification Point: 로그인 처리시간 5초 이내
```

test_login

로그인 기능의 정상 동작 확인

Verification Point

1. 회원가입 및 로그인 처리 시간이 5초 이내에 완료

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_prompt_engineering():
    start_timer()
    # Postman으로 프롬프트 기반 문제 생성 요청
    prompt_res = postman_call("POST", "/api/generate_problem_prompt", {"prompt": "Create a Level-3 Frontend problem with tests and example solution"})
    verify(prompt_res.status_code == 200)
    verify("input" in prompt_res.body and "output" in prompt_res.body)

    # 서버 측 신뢰성 평가 API 호출 (LLM이 생성한 문제를 서버에서 테스트)
    evaluate_res = postman_call("POST", "/api/evaluate_problem", {"problem_data": prompt_res.body})
    if evaluate_res.status_code != 200 or "fail" in evaluate_res.body:
        # 실패 시 대체 프롬프트 시도
        alt_res = postman_call("POST", "/api/generate_problem_prompt", {"prompt": "Alternative Level-3 Frontend problem"})
        verify(alt_res.status_code == 200)
        verify("input" in alt_res.body and "output" in alt_res.body)

        # 대체 프롬프트도 평가
        evaluate_alt_res = postman_call("POST", "/api/evaluate_problem", {"problem_data": alt_res.body})
        verify(evaluate_alt_res.status_code == 200)
        verify("success" in evaluate_alt_res.body)

    else:
        verify("success" in evaluate_res.body)

    elapsed = stop_timer()
    verify(elapsed <= 60s) # 1분 이내 평가 완료 검증

    # Python + pytest로 구조 검증
    run_pytest_script("test_prompt_structure.py")

# Manual Test: UI 상 생성된 문제 검토
# Verification Point: 문제 생성 및 평가 1분 이내, 실패 시 대체 프롬프트 처리 성공
```

test_prompt_engineering

프롬프트 실행 결과가 요구사항을 충족하고
예상대로 작동하는지 확인

Verification Point

- 문제 생성 및 검증 완료가 1분 이내에 완료되는지 확인
- 실패 시 대체 프롬프트로 재생성 후 동일 과정 반복

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_dashboard_access_and_profile():
    # 대시보드 접근
    hover_element("#user_id_display")
    verify_element_visible("#dashboard_button")
    click_element("#dashboard_button")
    wait_until_element_visible("#stats_graph", timeout=5s)
    verify_element_text_contains("#user_email_display", "moonsu99@example.com")

    # 프로필 사진 업로드
    click_element("#camera_icon")
    upload_file("#profile_upload_input", "path/to/test_image.jpg")
    click_element("#upload_confirm_button")
    verify_image_changed("#profile_picture")

    # 문제 탭 이동 후 문제 리스트 UI 확인
    click_element("#problems_tab")
    wait_until_element_visible("#problems_list", timeout=5s)

    # prompt_engineering 과정을 통해 문제 생성되었다 가정 (DB 반영 확인)
    # DB Query로 문제 확인
    db_result = db_query("SELECT * FROM problems WHERE difficulty='Level-3' AND topic='Frontend'")
    verify(db_result.count > 0)

    # Verification Point: 2분 이내 문제 생성(앞서 test_prompt_engineering 내 1분+추가 margin), DB일치성 검증
```

test_dashboard_access_and_profile
사용자가 요청한 주제와 난이도에 맞는 문제 생성
및 데이터 동기화 검증

Verification Point

- 문제 생성 및 검증 완료가 1분 이내에 완료되는지 확인
- 목록에 표시된 문제 정보가 데이터베이스와 일치

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_problem_filtering():
    navigate_to_url("https://webshooter.app")
    click_element("#전체실습문제_button")
    wait_until_element_visible("#problem_filter_area", timeout=5s)

    # 기술유형 및 난이도 선택 (Prompt로 생성된 Level-3 Frontend 문제 확인)
    click_element("#기술유형_select")
    click_element("//li[text()='Frontend']")
    click_element("#난이도_select")
    click_element("//li[text()='Level3']")

    verify_problem_list_filtered("Frontend", "Level3")
```

test_problem_filtering

전체 실습문제 페이지에서 필터링 및 UI 검증

Verification Point

1. 목록에 표시된 문제 정보가 데이터베이스와 일치

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_code_editor_live_preview():
    cypress_run("test_editor_response_time.spec.js")

    input_text("#code_editor", "%%%")
    click_element("#run_code_button")
    wait_until_element_visible("#error_message", timeout=5s)
    verify_element_text_contains("#error_message", "유효하지 않은 코드")

    # Manual Test: 실시간 미리보기 확인
    # Verification Point: 비정상 입력 처리
```

test_code_editor_live_preview 실시간 편집 및 미리보기 기능의 정상 작동 확인

Verification Point

1. 챗봇의 응답이 2분 이내에 반환되는지 확인
2. 비정상 입력에 대해 적절한 오류 메시지 반환

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_ai_chatbot():
    cypress_run("test_chatbot_autosave.spec.js")

    # Manual Test: 핫 리로딩/하이라이팅 확인
    # Verification Point: 자동 저장 코드 복원
```

test_ai_chatbot

챗봇의 핫 리로딩 및 자동 저장 기능 확인

Verification Point

- 자동 저장된 코드 복원 및 실시간 동작 여부

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_grading_panel():
    input_text("#code_submission", "function sum(a,b){return a+b;}")
    click_element("#submit_code_button")
    start_timer()
    wait_until_element_visible("#feedback_panel", timeout=5s)
    elapsed = stop_timer()
    verify(elapsed <= 5s)
    verify_element_text_contains("#feedback_panel", "테스트 통과")

    run_pytest_script("test_grading_feedback.py")

# Verification Point: 채점/피드백 5초 이내
```

test_grading_panel

코드 제출 후 피드백 및 채점 기능 검증

Verification Point

1. 채점 및 피드백 제공이 5초 이내에 완료되는지 확인

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_dashboard_update():
    navigate_to_url("https://webshooter.app/dashboard")
    old_count = get_element_text("#completed_problems_count")

    # 새로운 문제를 prompt로 추가 생성 (다른 난이도/주제)
    start_timer()
    alt_prompt_res = postman_call("POST", "/api/generate_problem_prompt", {"prompt": "Create a Level-2 Frontend problem"})
    verify(alt_prompt_res.status_code == 200)

    # 평가
    alt_eval = postman_call("POST", "/api/evaluate_problem", {"problem_data": alt_prompt_res.body})
    if alt_eval.status_code != 200 or "fail" in alt_eval.body:
        # 대체 프롬프트
        alt_res2 = postman_call("POST", "/api/generate_problem_prompt", {"prompt": "Alternative Level-2 Frontend problem"})
        verify(alt_res2.status_code == 200)
        alt_eval2 = postman_call("POST", "/api/evaluate_problem", {"problem_data": alt_res2.body})
        verify(alt_eval2.status_code == 200)
        verify("success" in alt_eval2.body)
    else:
        verify("success" in alt_eval.body)

    elapsed = stop_timer()
    verify(elapsed <= 60s)

    # 대시보드 갱신 확인
    wait(120s) # 문제 반영 대기
    refresh_page()
    new_count = get_element_text("#completed_problems_count")
    verify(int(new_count) > int(old_count))

    # DB 비교
    db_result_count = db_query("SELECT COUNT(*) FROM completed_problems WHERE user_id='moonsu99'")
    verify(int(new_count) == db_result_count)

# Verification Point: 데이터 업데이트 5초 이내(새로고침 후), DB 일치성
```

test_dashboard_update

대시보드 업데이트 및 시각화 데이터의 정확성 검증

Verification Point

- 데이터 업데이트가 5초 이내에 완료되는지 확인
- 시각화된 정보가 데이터베이스 기록과 정확히 일치하는지 확인

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_problem_solving_and_sharing():
    # 문제 페이지 접근 (이미 prompt로 생성)
    navigate_to_url("https://webshooter.app/problems")
    click_element(".problem_item[data-level='Level3'][data-topic='Frontend']")
    wait_until_element_visible("#problem_description_section", timeout=120s)
    verify_element_visible("#code_editor")
    verify_element_visible("#live_preview")

    # 제출 후 공유, 댓글, 피드백 시나리오
    input_text("#code_editor", /* 문제 요구사항에 맞는 코드 작성 */)
    click_element("#submit_solution_button")
    wait_until_element_visible("#feedback_panel", timeout=5s)
    verify_element_text_contains("#feedback_panel", "유사도")

    # 유사도 95% 이상 가정 → 공유하기 버튼
    click_element("#share_solution_button")
    wait_until_element_visible("#share_popup", timeout=5s)
    verify_element_text_contains("#share_popup_message", "공유하시겠습니까?")
    click_element("#share_confirm_button")
    wait_until_element_visible("#board_section", timeout=5s)
    verify_element_text_contains("#board_submission_info", "moonsu99")

    # 댓글
    input_text("#comment_input", "잘 풀었네요!")
    click_element("#comment_submit_button")
    verify_element_text_contains("#comments_list", "잘 풀었네요!")

    # 라인 피드백
    click_element(".line_number[data-line='10']")
    input_text("#line_feedback_input", "이 부분 로직 최적화 가능")
    click_element("#line_feedback_submit")
    verify_element_text_contains("#line_feedback_panel", "이 부분 로직 최적화 가능")
```

test_problem_solving_and_sharing
문제풀이 후 문제 공유 기능 작동 및 이상성 검증

Verification Point

1. 해당 이벤트마다 기능 정상 작동하는지 확인

03 / TEST DESIGN

Release Testing - End to End Testing

```
function test_admin_functions():
    hover_element("#user_id_display")
    click_element("#logout_button")
    wait_until_element_visible("#login_email_field", timeout=5s)

    input_text("#login_email_field", "admin@example.com")
    input_text("#login_password_field", "AdminPass123!")
    click_element("#login_submit_button")
    wait_until_element_visible("#admin_dashboard", timeout=5s)

    navigate_to_url("https://webshooter.app/admin/users")
    wait_until_element_visible("#user_list", timeout=5s)

    verify_element_text_contains(".user_item[data-user='moonsu99']", "moonsu99@example.com")
    verify_element_text_contains(".user_item[data-user='moonsu99']", "생성 문제 수")
    verify_element_text_contains(".user_item[data-user='moonsu99']", "생성 댓글 수")

    click_element(".user_item[data-user='moonsu99']")
    wait_until_element_visible("#user_profile", timeout=5s)
    verify_element_text_contains("#user_profile", "moonsu99@example.com")

    wait_until_element_visible(".comment_item:contains('잘 풀었네요!')", timeout=5s)
    click_element(".delete_comment_button[data-comment='잘 풀었네요!']")
    verify_element_not_present(".comment_item:contains('잘 풀었네요!')")
```

test_admin_functions

관리자 기능 작동 여부 및 이상성 검증

Verification Point

- 해당 이벤트마다 관리자 기능 정상 작동하는지 확인

Performance Testing 02

03 / TEST DESIGN

Release Testing - Performance Testing

```
function test_concurrent_users():
    # Tools: JMeter(부하), Locust(부하), AWS CloudWatch(모니터링)

    # Automated Test (JMeter 사용)
    jmeter_result = run_jmeter_test("jmeter_scripts/200_users_test.jmx")
    verify_response_time(jmeter_result, <=2s)
    verify_no_bottleneck(jmeter_result)

    # Locust로 사용자 요청(챗봇 질문, 문제 생성) 시뮬레이션
    locust_result = run_locust_test("locust_scripts/50_users_scenario.py")
    verify_response_time(locust_result, <=2s)
    verify_no_bottleneck(locust_result)

    # Verification Points:
    # 응답 시간 2초 이내, 병목 현상 없음
```

test_concurrent_users

동시 사용자 접속 및 요청 처리 성능 확인

Verification Point

1. 서버 응답 시간이 2초 이내로 유지되는지 확인
2. 동시 요청 처리에서 병목 현상 없는지 확인

03 / TEST DESIGN

Release Testing - Performance Testing

```
function test_host_client_communication():
    # Tools: Wireshark, Browser DevTools

    # Automated:
    # Selenium으로 요청 전송
    click_element("#request_data_button")
    wait_until_element_visible("#response_data", 5s)
    verify_data_integrity("#response_data")

    # WebSocket 종료 시나리오
    simulate_websocket_close()
    verify_no_data_loss()

    # Manual:
    # DevTools Network 탭 확인, Wireshark로 패킷 분석
    # Verification: 전송 속도 10Mbps 이상, 소켓 종료 시 데이터 손실 없음
```

test_host_client_communication

클라이언트와 서버 간 데이터 통신의 정확성과 신뢰성 확인

Verification Point

1. 전송 속도 10Mbps 이상
2. 소켓 종료 시 데이터 손실 없음

03 / TEST DESIGN

Release Testing - Performance Testing

```
function test_monitoring_server_performance():
    # Tools: AWS CloudWatch, Grafana, Selenium

    # Automated:
    refresh_page()
    verify_ui_sync(<=100ms delay) # UI에서 데이터 갱신 확인

    # Manual:
    # Grafana 대시보드에서 서버 부하, 동기화 상태 검토
    # Verification: 동기화 지연 100ms, 갱신 주기 1초 이내
```

test_monitoring_server_performance
실시간 동기화 및 서버 처리 성능 확인

Verification Point

1. 동기화 지연이 100ms 이내인지 확인
2. 데이터 갱신 주기가 1초 이내인지 확인

03 / TEST DESIGN

Release Testing - Performance Testing

```
function test_page_load_time():
    # Tools: Lighthouse, Selenium

    # Automated:
    lighthouse_result = run_lighthouse("https://webshooter.app")
    verify_page_load(lighthouse_result, <=3s)

    # Manual:
    # 다양한 브라우저, 디바이스, 느린 네트워크 환경에서 확인
    # Verification: 3초 이내로딩 완료
```

test_page_load_time

페이지 로딩 시간이 성능 요구사항을 충족하는지 확인

Verification Point

1. 페이지 로딩 시간이 3초이내인지 확인

03 / TEST DESIGN

Release Testing - Performance Testing

```
function test_data_save_load_time():
    # Tools: Selenium, SQLAlchemy

    # Automated:
    save_start = start_timer()
    click_element("#save_data_button")
    wait_until_element_visible("#save_complete_msg", timeout=5s)
    save_elapsed = stop_timer()
    verify(save_elapsed <= 5s)

    # Manual:
    # UI에서 데이터 로드 확인, DB에서 정확성 검토
    # Verification: 저장/로드 5초 이내
```

test_data_save_load_time
학습 데이터 저장 및 불러오기 속도 확인

Verification Point

- 저장 및 불러오기 작업 완료가 5초 이내인지 확인

03 / TEST DESIGN

Release Testing - Performance Testing

```
function test_monitoring_server_performance():
    # Tools: AWS CloudWatch, Grafana, Selenium

    # Automated:
    refresh_page()
    verify_ui_sync(<=100ms delay) # UI에서 데이터 갱신 확인

    # Manual:
    # Grafana 대시보드에서 서버 부하, 동기화 상태 검토
    # Verification: 동기화 지연 100ms, 갱신 주기 1초 이내
```

test_monitoring_server_performance
실시간 동기화 및 서버 처리 성능 확인

Verification Point

1. 동기화 지연 100ms 이내
2. 갱신 주기 1초 이내

03 / TEST DESIGN

Release Testing - Performance Testing

```
function test_admin_page_performance():
    # Tools: Selenium, pgAdmin(DB Query)

    # Automated:
    start_timer()
    click_element("#admin_users_tab")
    wait_until_element_visible("#user_list", 5s)
    elapsed = stop_timer()
    verify(elapsed <= 5s)

    # 사용자 삭제/수정 작업 시간 측정
    start_timer()
    click_element(".delete_user_button[data-user='testuser']")
    wait_until_element_not_present(".user_item[data-user='testuser']", 5s)
    elapsed_delete = stop_timer()
    verify(elapsed_delete <= 5s)

    # DB 확인
    db_check = db_query("SELECT * FROM users WHERE username='testuser'")
    verify(db_check.count == 0)

    # Verification: 모든 작업 5초 이내 처리
```

test_admin_page_performane
관리자 페이지의 데이터 처리 성능을 검증

Verification Point

1. 실행 결과 반영이 5초 이내에 이루어지는지 확인

03 Compatibility Testing

03 / TEST DESIGN

Release Testing - Compatibility Testing

```
function test_browser_compatibility():
    # 1. Edge, Firefox 기능 검증
    # Tools: BrowserStack, Selenium WebDriver, Lighthouse
    # BrowserStack 상에서 Edge, Firefox 환경 생성 가정

    browsers = ["Edge", "Firefox"]
    for browser in browsers:
        browserstack_session = start_browserstack_session(browser)
        navigate_to_url("https://webshooter.app")
        # 로그인 기능 테스트
        input_text("#login_email_field", "testuser@example.com")
        input_text("#login_password_field", "ValidPass123!")
        click_element("#login_submit_button")
        wait_until_element_visible("#dashboard", timeout=5s)
        verify_element_visible("#dashboard")

        # 문제 생성 등 주요 기능 검증
        click_element("#problems_tab")
        wait_until_element_visible("#problems_list", timeout=5s)
        verify_element_visible(".problem_item")

        # 종료
        close_browserstack_session(browserstack_session)

    # Lighthouse를 통한 Chrome 렌더링 검사 (개념적)
    lighthouse_result = run_lighthouse("https://webshooter.app")
    verify_no_rendering_issues(lighthouse_result)

    # Validation Points:
    # 모든 브라우저에서 동일한 UI
```

test_browser_compatibility

WebShooter의 브라우저 호환성 점검

Validation Point

- 모든 브라우저에서 UI가 동일하게 렌더링되는지 확인
- 브라우저 간 기능 차이 없음

03 / TEST DESIGN

Release Testing - Compatibility Testing

```
function test_api_and_error_messages():
    # Tools: Postman, Selenium
    # Postman으로 브라우저별 API 요청 테스트
    response_edge = postman_call("GET", "/api/problems?browser=edge")
    response_firefox = postman_call("GET", "/api/problems?browser=firefox")

    verify(response_edge.status_code == 200)
    verify(response_firefox.status_code == 200)
    verify(response_edge.body == response_firefox.body) # 브라우저별 응답 동일성 확인

    # Selenium으로 네트워크 요청 실패 상황 시뮬레이션
    simulate_network_failure()
    click_element("#refresh_problems_button")
    wait_until_element_visible("#error_message", timeout=5s)
    verify_element_text_contains("#error_message", "네트워크 오류 발생")

    # Validation Points:
    # API 요청 결과 브라우저 간 동일, 네트워크 실패 시 사용자 오류 메시지
```

test_api_and_error_messages

WebShooter의 API 호환성 점검

Validation Point

1. API 요청 결과가 브라우저 간 동일
2. 네트워크 실패 시 사용자 친화적인 오류 메시지가 표시

03 / TEST DESIGN

Release Testing - Compatibility Testing

```
function test_javascript_disabled():
    # Tools: Browser DevTools, Selenium
    # JavaScript 비활성화: 개발자 도구 사용(개념)
    disable_javascript_via_devtools()

    navigate_to_url("https://webshooter.app")
    # Selenium으로 JS 비활성 상태에서 페이지 로드 검증
    wait_until_element_visible("#js_disabled_message", timeout=5s)
    verify_element_text_contains("#js_disabled_message", "JavaScript 비활성화로 기능 제한")

    # Validation Points:
    # JS 비활성 시 경고 메시지, 제한된 기능 안내 정확히 표시
```

test_javascript_disabled

JavaScript 비활성화 기능 검증

Validation Point

1. JavaScript 비활성화 시 명확한 경고 메시지가 표시
2. 제한된 기능 안내가 정확히 제공

03 / TEST DESIGN

Release Testing - Compatibility Testing

```
function test_min_spec_devices():
    # Tools: Real Device Testing, BrowserStack
    # BrowserStack: 저사양 환경(1.0GHz, 2GB RAM) 시뮬레이션
    low_spec_session = start_browserstack_session("Chrome", cpu="1GHz", ram="2GB")
    navigate_to_url("https://webshooter.app")
    verify_page_responsiveness(low_spec_session)

    # 실제 저사양 기기에서 테스트 (Manual)
    # manual_check_on_real_device()

    # Validation: 저사양에서도 기능 정상 동작
```

test_min_spec_devices 최소 사양 환경 검증

Validation Point

1. 저사양 기기에서 주요 기능이 정상 작동

03 / TEST DESIGN

Release Testing - Compatibility Testing

```
function test_high_resolution_display():
    # Tools: Figma Mirror, BrowserStack
    high_res_session = start_browserstack_session("Chrome", resolution="4K")
    navigate_to_url("https://webshooter.app")

    # manual_compare_with_figma()

    # Verify no scaling issues
    verify_no_scaling_issues(high_res_session)

    # Validation: 고해상도에서 UI 정확히 표시
```

test_high_resolution_display 고해상도 디스플레이 검증

Validation Point

1. 고해상도 환경에서도 UI가 정확히 배치
2. 텍스트 및 이미지가 스케일링 문제 없이 표시

03 / TEST DESIGN

Release Testing - Compatibility Testing

```
function test_responsive_design():
    # Tools: Chrome DevTools, Selenium Grid
    # Chrome DevTools를 통한 화면 비율 변경 (16:9, 4:3, 3:2)
    aspect_ratios = ["16:9", "4:3", "3:2"]
    for ratio in aspect_ratios:
        set_viewport_ratio(ratio)
        navigate_to_url("https://webshooter.app")
        verify_layout_integrity()

    # Selenium Grid를 통한 다양한 디바이스 해상도 테스트
    resolutions = ["1920x1080", "1280x720", "1024x768"]
    for res in resolutions:
        grid_session = start_selenium_grid_session(resolution=res)
        navigate_to_url("https://webshooter.app")
        verify_no_ui_breaks(grid_session)
        end_selenium_grid_session(grid_session)

    # Validation: 모든 해상도/비율에서 UI 깨짐 없음, 반응형 레이아웃 정상 적용
```

test_responsive_design 반응형 설계 검증

Validation Point

- 모든 화면 비율에서 UI가 깨지지 않고 정확히 표시
- 레이아웃이 각 해상도에 맞게 조정

03 / TEST DESIGN

TESTCASE

Security Testing 04

03 / TEST DESIGN

Release Testing - Security Testing

```
function test_https_protocol():
    # Tools: Browser DevTools, OpenSSL
    open_chrome_devtools()
    verify_https_enabled("https://webshooter.app")

    openssl_result = run_openssl("openssl s_client -connect webshooter.app:443")
    verify_certificate_validity(openssl_result)
    verify_tls_protocol_version(openssl_result, "TLS1.2 or higher")

    # Verification: 모든 전송 HTTPS, 인증서 유효 & 최신 프로토콜 사용
```

test_https_protocol

HTTPS 프로토콜 검증

Verification Point

1. 모든 데이터 전송이 HTTPS를 통해 암호화
2. SSL/TLS 인증서가 유효하며 최신 암호화 프로토콜
(예: TLS 1.2/1.3)을 사용

03 / TEST DESIGN

Release Testing - Security Testing

```
function test_password_policy():
    # Tools: Selenium
    open_browser("Chrome")
    navigate_to_url("https://webshooter.app/signup")

    # 짧은 비밀번호 테스트 (비정상)
    input_text("#signup_email_field", "testweak@example.com")
    input_text("#signup_password_field", "short")
    click_element("#signup_submit_button")
    verify_element_text_contains("#error_message", "비밀번호는 8자 이상")

    # 특수문자 없는 단순 비밀번호 테스트
    input_text("#signup_email_field", "testweak2@example.com")
    input_text("#signup_password_field", "NoSpecialChar123")
    click_element("#signup_submit_button")
    verify_element_text_contains("#error_message", "특수문자 포함 필요")

    close_browser()
    # Verification: 정책 불만족 시 명확한 오류 안내
```

test_password_policy 비밀번호 정책 검증

Verification Point

- 비밀번호 최소 길이(예: 8자 이상) 및 문자 조합(대소문자, 숫자, 특수문자) 요구
- 오류 메시지가 명확하고 사용자에게 가이드 제공

03 / TEST DESIGN

Release Testing - Security Testing

```
function test_session_expiration():
    # Tools: Selenium, DevTools
    open_browser("Chrome")
    login("testuser@example.com", "ValidPass123!")
    simulate_inactivity(30min)
    refresh_page()
    verify_current_url_contains("/login") # 자동 로그아웃 확인

    # 세션 토큰 변경 후 요청
    devtools_set_cookie("session_id", "tampered_session_token")
    navigate_to_url("https://webshooter.app/dashboard")
    verify_element_text_contains("#error_message", "세션 유효하지 않음")

close_browser()
# Verification: 세션 만료, 변경 시 접속 차단
```

test_session_expiration

장시간 미활동 후 세션 만료 및 비정상 세션 재활용 확인

Verification Point

1. 세션 만료 후 새로고침 시 자동 로그아웃
2. 비정상적인 세션 재활용 시 오류 메시지 제공

03 / TEST DESIGN

Release Testing - Security Testing

```
function test_admin_page_security():
    # Tools: OWASP ZAP, Splunk
    zap_result = run_owasp_zap_scan("https://webshooter.app/admin")
    verify_no_critical_vulnerabilities(zap_result)

    # Splunk 로그 모니터링: 비정상 접근 감지
    verify_alert_generated_for("abnormal_admin_access")

    # Verification: 관리자 페이지 비정상 접근 시도 감지 및 알림
```

test_admin_page_security

관리자 페이지 비정상 접근, 로그 감시 확인

Verification Point

1. 모든 관리자 페이지 접근이 로그로 기록
2. 비정상적인 접근 시 관리자에게 알림 발송

05 Failover and Recovery Testing

03 / TEST DESIGN

Release Testing - Failover and Recovery Testing

```
function test_alerting_system():
    # Tools: AWS SNS, Email Notification Tools
    # 서버 다운타임 시 관리자에게 알림 전송 확인

    simulate_server_downtime("webshooter-app-instance-2")
    # SNS 토픽으로 알림 발송 가정
    verify_email_received("admin@example.com", subject_contains="서버 다운")

    # 알림 메시지 내용 확인: 복구 예상 시간, 상태 정보
    verify_email_body_contains("admin@example.com", "복구 예상 시간")

    # Verification: 장애 발생 시 알림 전송 및 상태 정보 포함
```

test_alerting_system

AWS 자동 스케일링 복구 테스트

Verification Point

- 장애 발생 후 5분 내 새로운 서버 인스턴스 생성 및 정상화
- 서비스 중단 시간 동안 모든 요청이 적절히 처리(또는 오류 메시지 제공)

03 / TEST DESIGN

Release Testing - Failover and Recovery Testing

```
function test_emergency_recovery_procedure():
    # Tools: Manual Testing, Incident Response Docs
    # 문서화된 복구 절차에 따라 DB 및 서버 복구

    # Manual step: 문서에 따른 복구 스크립트 실행 (개념적)
    run_recovery_script("db_restore.sh")
    run_recovery_script("server_redeploy.sh")

    # 로그 기록 확인
    verify_logs_for("recovery completed")

    # 복구 후 서비스 접근
    response = make_http_request("https://webshooter.app")
    verify(response.status_code == 200)

    # Verification: 문서화된 절차대로 복구 가능, 서비스 정상화
```

test_emergency_recovery_procedure 비상 복구 절차 검증

Verification Point

1. 복구 절차가 문서화된 대로 실행 가능
2. 복구 이후 모든 서비스가 정상 작동

03 / TEST DESIGN

Release Testing - Failover and Recovery Testing

```
function test_data_loss_prevention():
    # Tools: AWS RDS Snapshot, SQLAlchemy
    # 장애 상황에서 DB 백업 복원

    simulate_db_failure()
    restore_db_from_snapshot("webshooter_snapshot")
    db_result = db_query("SELECT * FROM user_learning_records WHERE user_id='testuser'")
    verify(db_result.count > 0)

    # Verification: 데이터 손실 없음, 장애 전 기록 복원
```

test_data_loss_prevention

데이터 유실 방지 검증

Verification Point

- 저장된 데이터가 유실되지 않고 복구 후에도 그대로 반영

03 / TEST DESIGN

Release Testing - Failover and Recovery Testing

```
function test_data_integrity_after_failure():
    # Tools: Selenium, Manual Testing
    # 문제 생성/학습 기록 저장 중 장애 발생 후 데이터 확인

    open_browser("Chrome")
    login("testuser@example.com", "ValidPass123!")
    input_text("#new_problem_title", "Test Problem")
    click_element("#save_problem_button")

    # 문제 생성 도중 장애 발생
    simulate_server_downtime("webshooter-app-instance-4")
    wait(5s) # 장애 발생 시간
    simulate_server_recovery("webshooter-app-instance-4")

    # 복구 후 페이지 재접속
    refresh_page()
    verify_element_text_contains(".problem_item", "Test Problem")

    close_browser()
    # Verification: 장애 발생 전 저장 요청 데이터 복구 후 반영
```

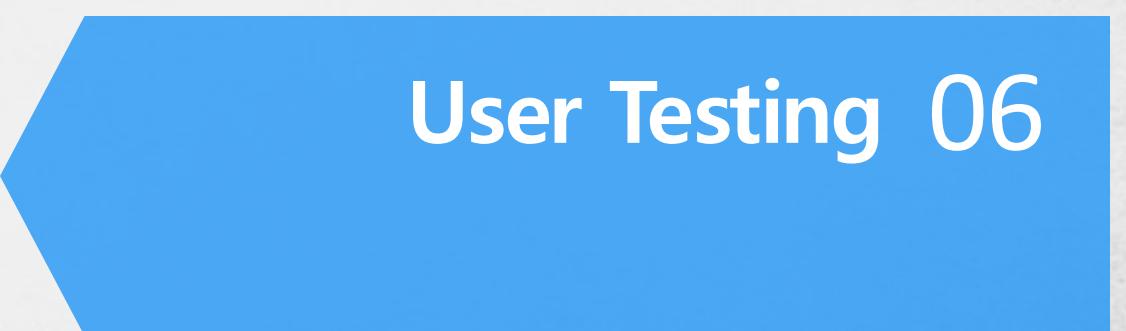
test_data_integrity_after_failure
중단 없는 사용자 경험 검증

Verification Point

1. 장애 발생 전 저장 요청 데이터 복구 후 반영

03 / TEST DESIGN

TESTCASE



User Testing 06

03 / TEST DESIGN

User Testing - Interaction Testing

```
function test_drag_accuracy():
    open_browser("Chrome")
    navigate_to_url("https://webshooter.app/dragdrop_test")

    # Manual Test:
    # 1. 지정된 영역 내 드래그 시도
    # 2. 화면 경계 바깥으로 드래그 시도
    # 3. 여러 번 드래그 실행 후 위치 오차 측정 (10회 평균 오차)

    # Pseudocode 개념:
    element_to_drag = find_element("#draggable_item")
    valid_target = find_element("#valid_drop_area")
    invalid_target = find_element("#outside_area")

    for i in 1 to 10:
        drag_and_drop(element_to_drag, valid_target)
        distance_error = measure_drop_position_error(element_to_drag)
        verify(distance_error <= 5px)

    # 비정상 값(화면 바깥)
    drag_and_drop(element_to_drag, invalid_target)
    verify_element_text_contains("#error_message", "드래그 벗어남")

close_browser()

# Validation: 드래그 앤 드롭 동작 정확도 5px 이내
```

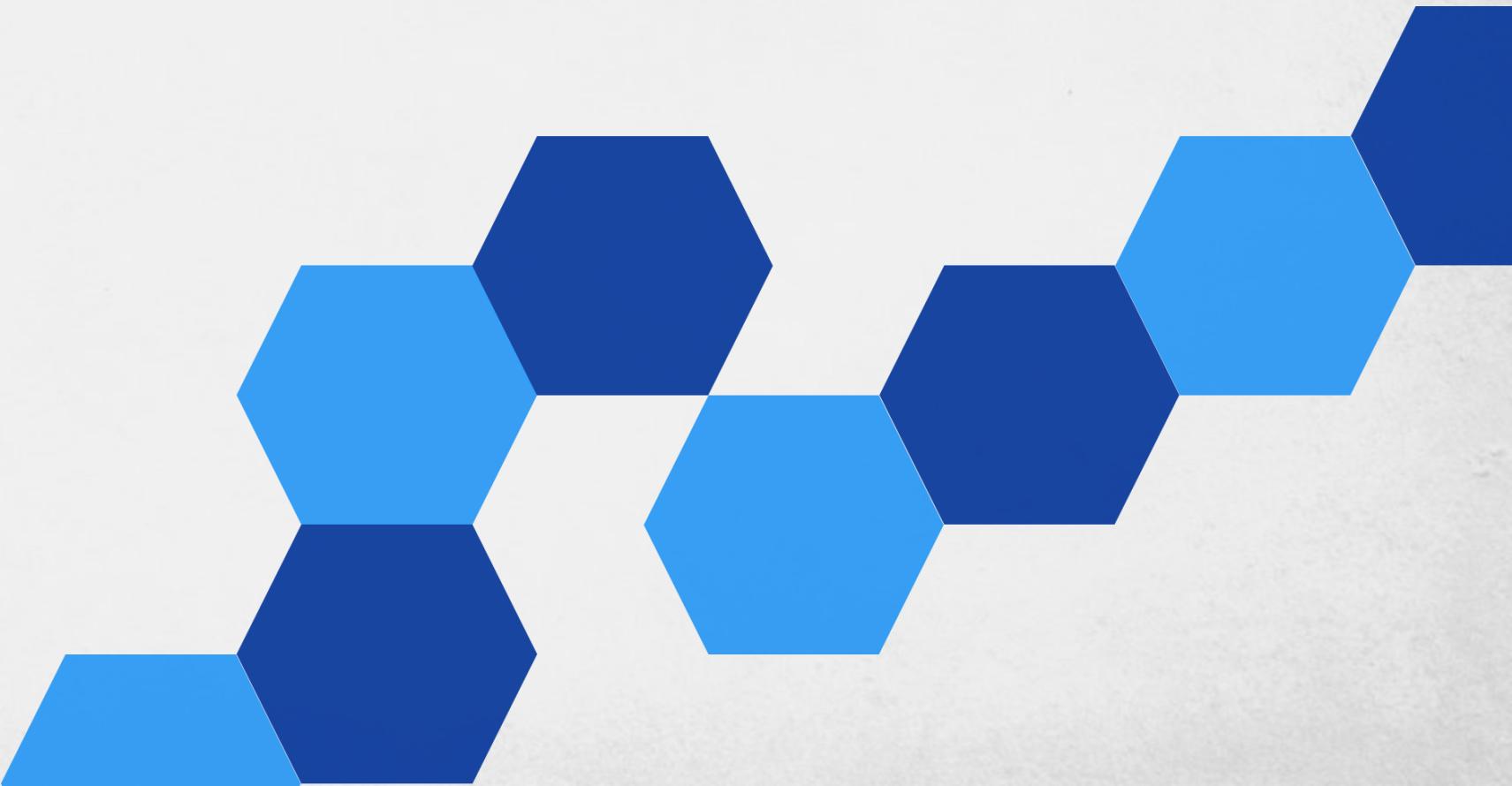
test_drag_accuracy

사용자의 마우스에 의한 입력이 시스템에 정확히 반영되는지 확인

Verification Point

1. 드래그 앤 드롭 동작이 정확히 수행

감사합니다



Q&A / Prompt Engineering

Real Example

```
from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {
            "role": "system",
            "content": "You are an assistant that generates front-end web development coding challenges of intermediate difficulty. You will be given instructions about the problem to create, including requirements and context. Your response should include a clear problem statement, requirements, and any constraints or hints for the user. The problem should focus on HTML, CSS, and/or JavaScript, and test the user's ability to implement web features."
        },
        {
            "role": "user",
            "content": "Create an intermediate-level front-end coding challenge that involves building a responsive navigation bar with a dropdown menu. The requirements: it should adapt to mobile screens, include a hover-based dropdown on desktop, and a toggle menu on mobile. Include any relevant constraints and hints."
        }
    ],
    temperature=0.8,
    max_tokens=256,
    top_p=1
)
```