

LOL 소환사 분석 및 도전과제

2025114442황보현준 / 2025113197이주현

2025.12.17

Contents

01 개발 목적
02 초기 구상
03 핵심 기능 구현
04 부가 기능 추가

05 핵심 코드
06 주차 별 진행 과정
07 개인 별 기여 내용
08 결론과 개선점

기존의 전적 사이트의 문제

- 1. 너무 많은 정보가 표시됨(전적, 플레이한 챔피언, 숙련도,같이 플레이한 소한사 등)
- 2. 위의 문제로 보고자 하는 정보를 찾기가 힘들
- 3. 도전과제 등의 찾고 싶은 정보를 제공하지 않음





간결한 디자인

기존 사이트에서 표시되는
광고나 크게 중요하지 않은
정보들 없이 간결한 UI로 디자인



필요한 정보만 표시

필요 정보를 크게 보여주고
챔피언 초상화와 도전과제
아이콘, 그래프를 사용하여
데이터 시각화

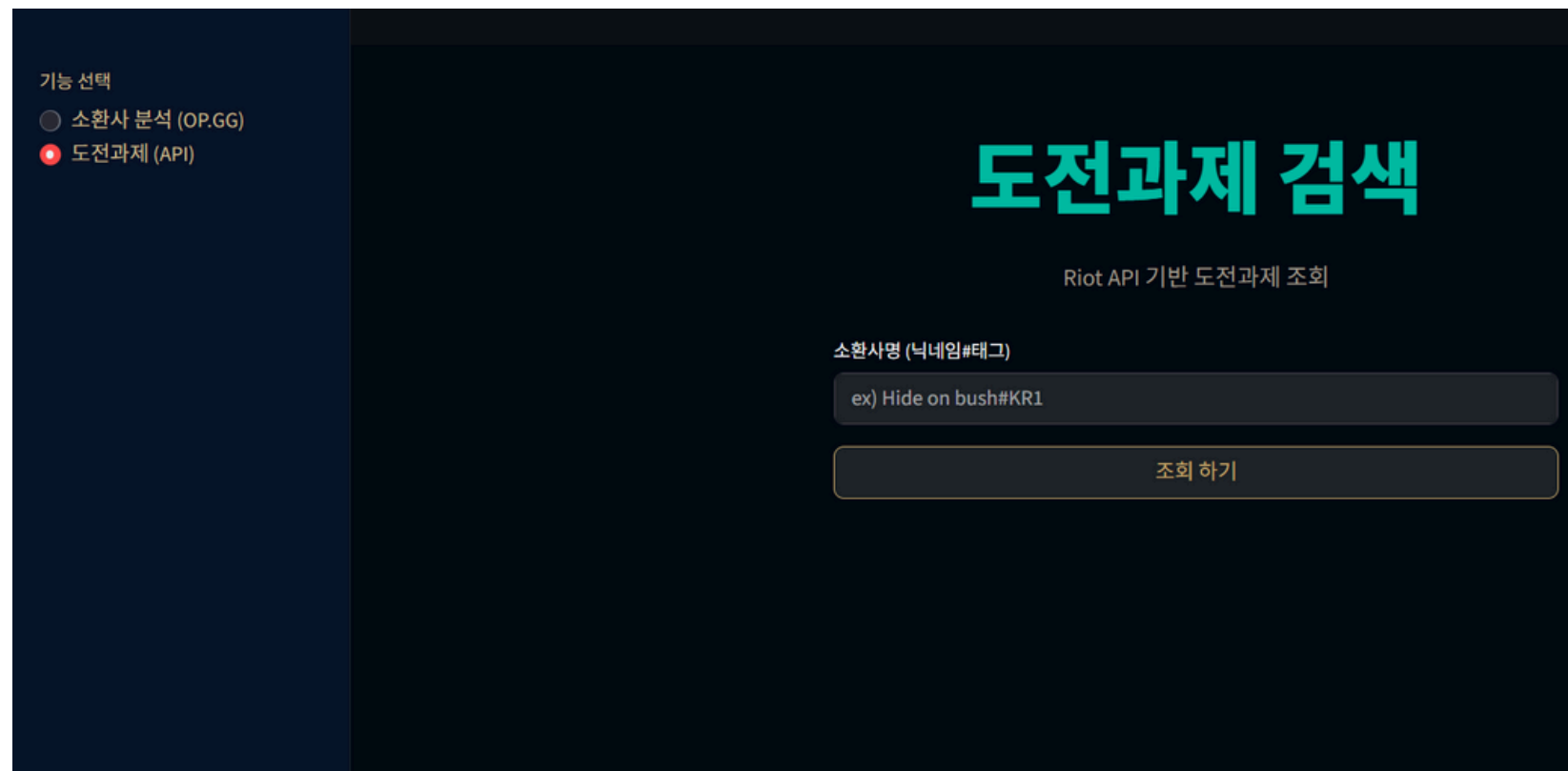
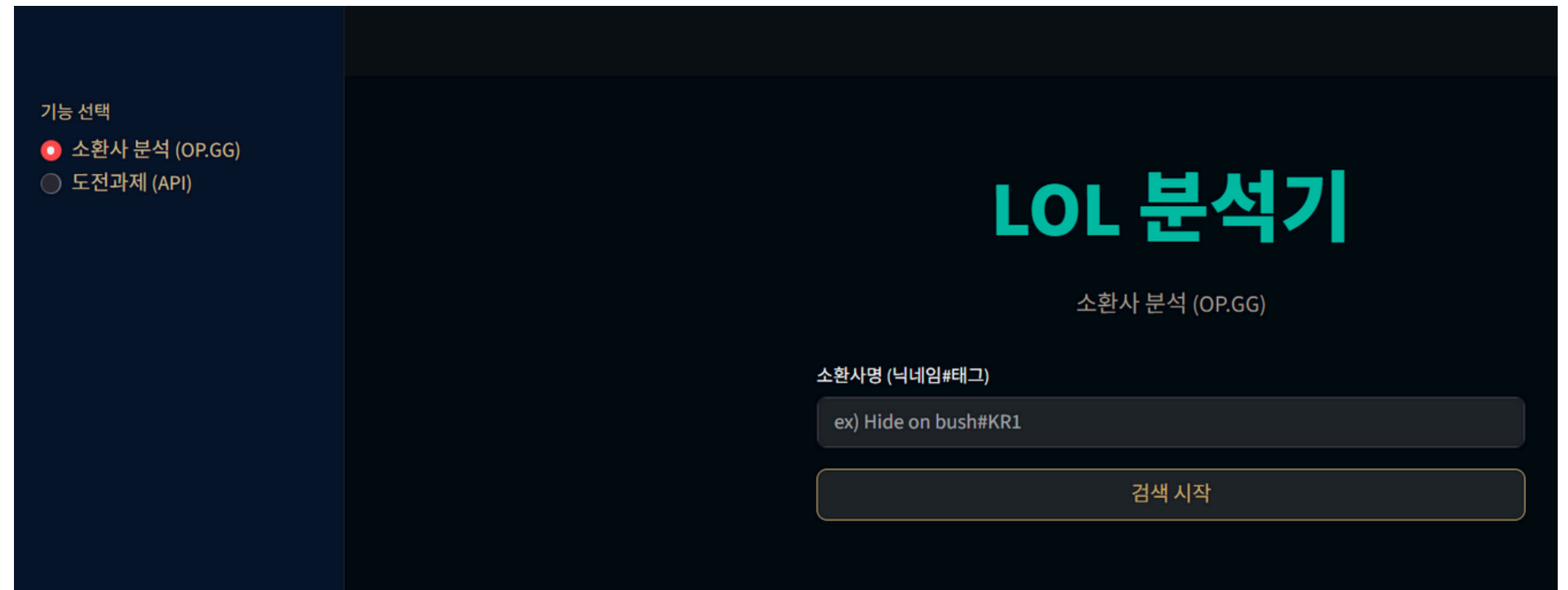


개인별 도전과제 정보 제공

대부분의 사이트들에서
제공되지 않는 도전과제
정보들을 시각화하여 제공

메인 검색창

닉네임#태그 형식으로 검색

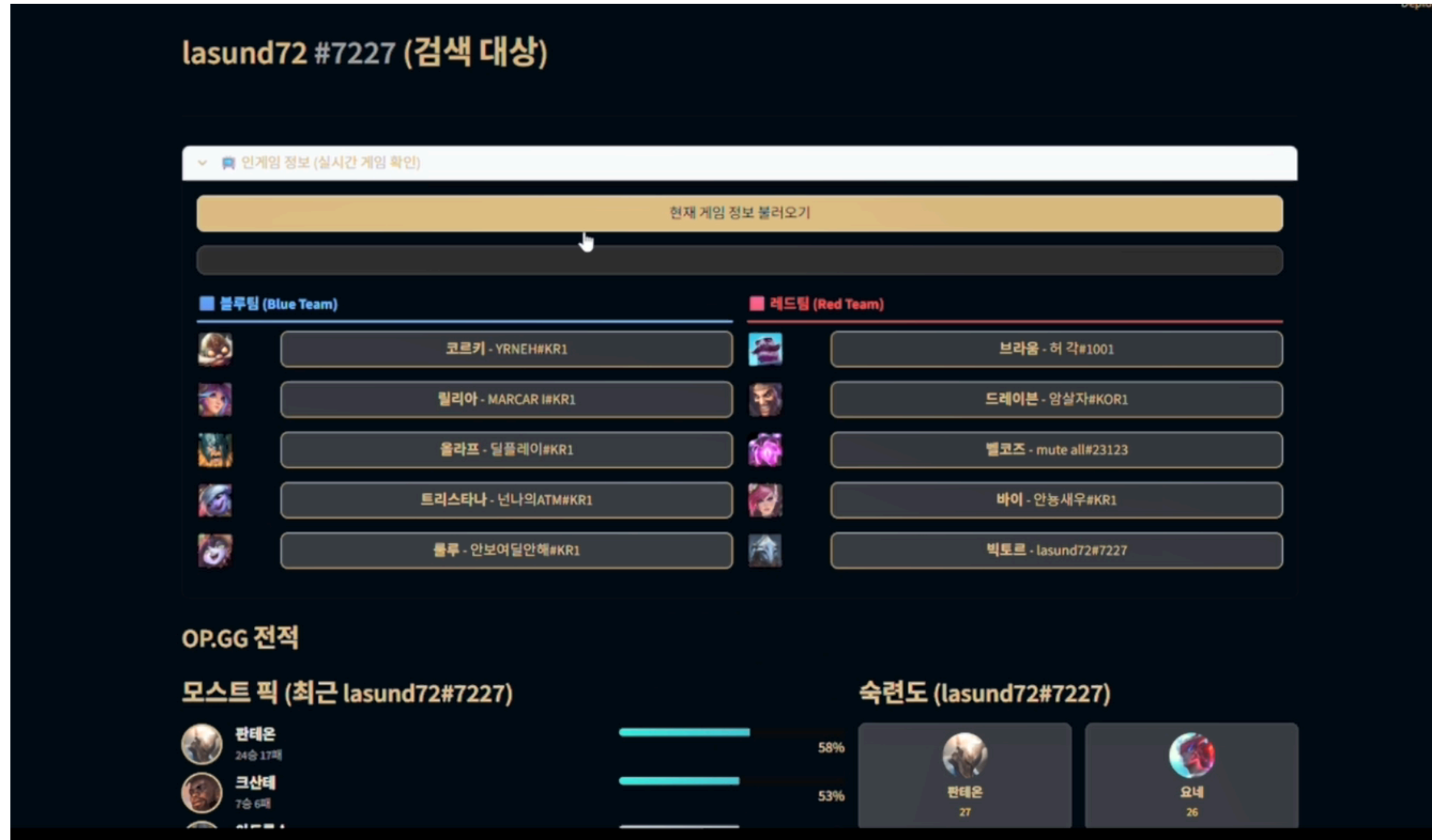


사이드 바의 메뉴 변경시
검색창 타이틀 변경



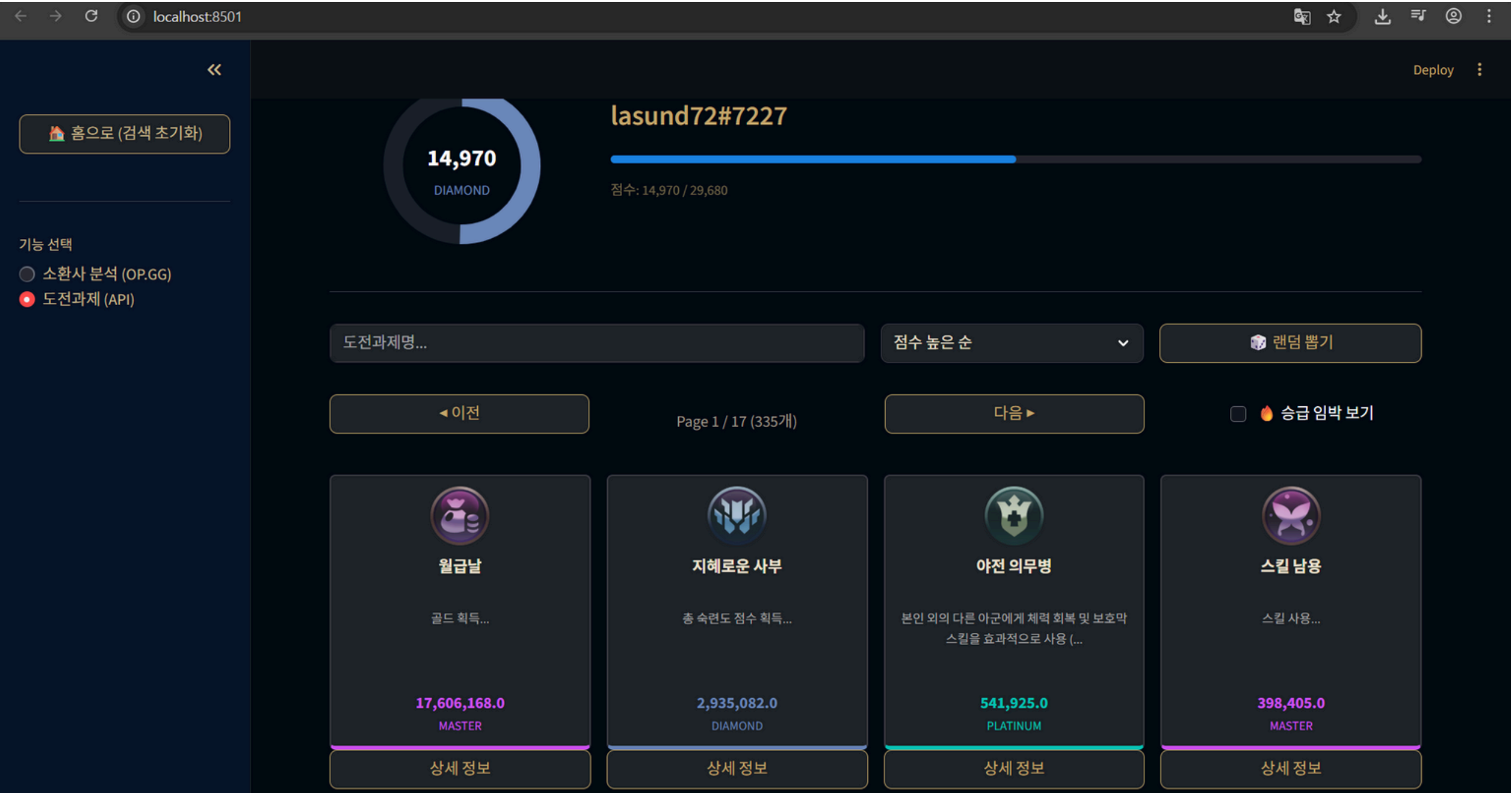
소환사 분석 탭

- 모스트 픽과 해당 챔피언 승률 막대그래프로 표시
- 가장 높은 숙련도를 가진 챔피언 5개의 레벨 표시
- 사이드바 상단의 버튼을 통해 초기 화면으로 이동



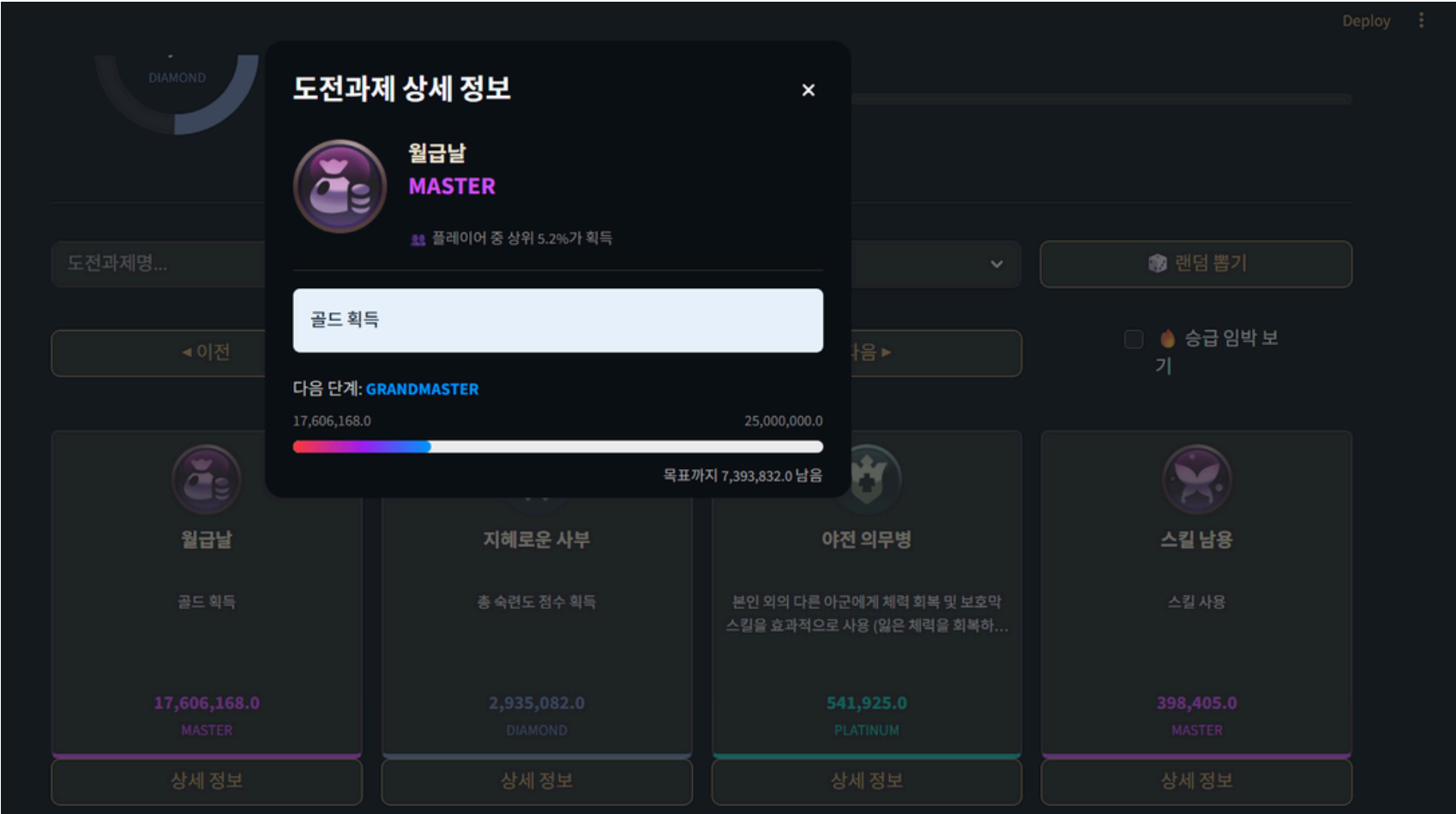
소환사 분석 탭

- 게임을 플레이 할 경우 인게임 정보 표시
- 아군 소환사, 적 소환사, 각 챔피언 모두 출력
- 닉네임을 누르면 해당 소환사의 검색창으로 이동



도전과제 검색 탭

- 화면 상단에 소환사의 도전과제 단계 표시
- 아래쪽에는 도전과제 별 달성 현황 표시

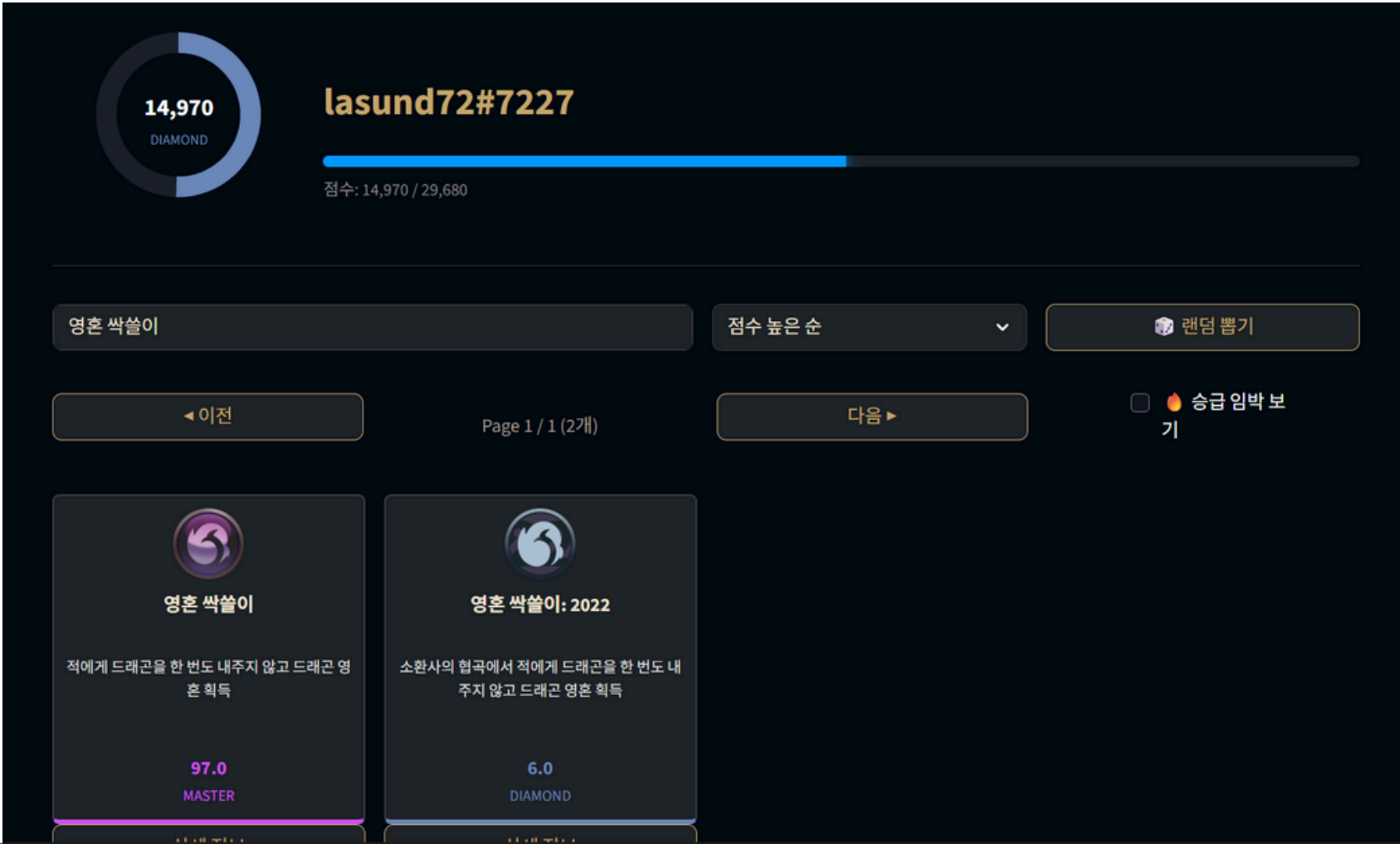


상세 정보

각 도전과제 밑에 있는 ‘상세 정보’ 버튼을 통해
도전과제별 상세 정보 표시

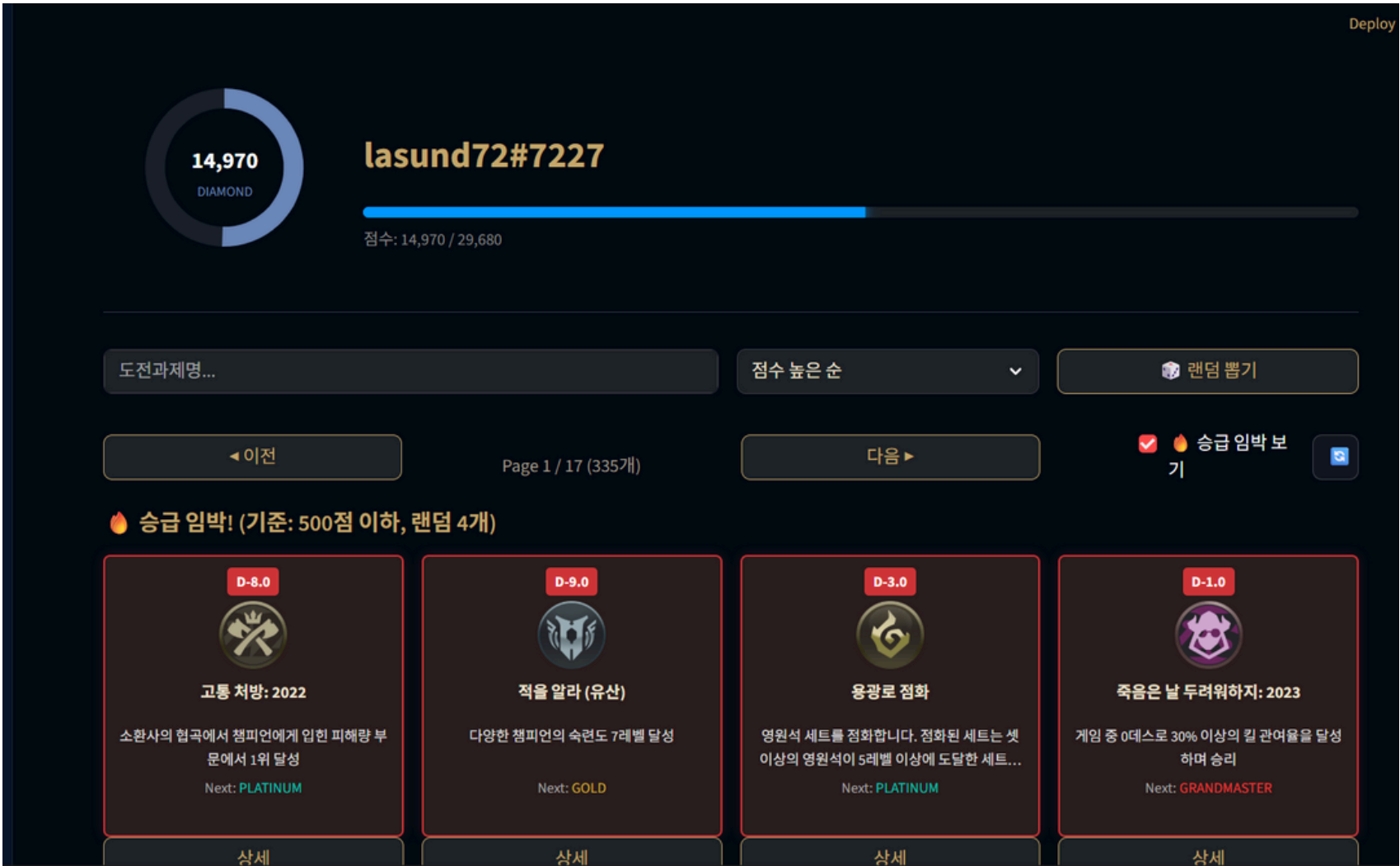
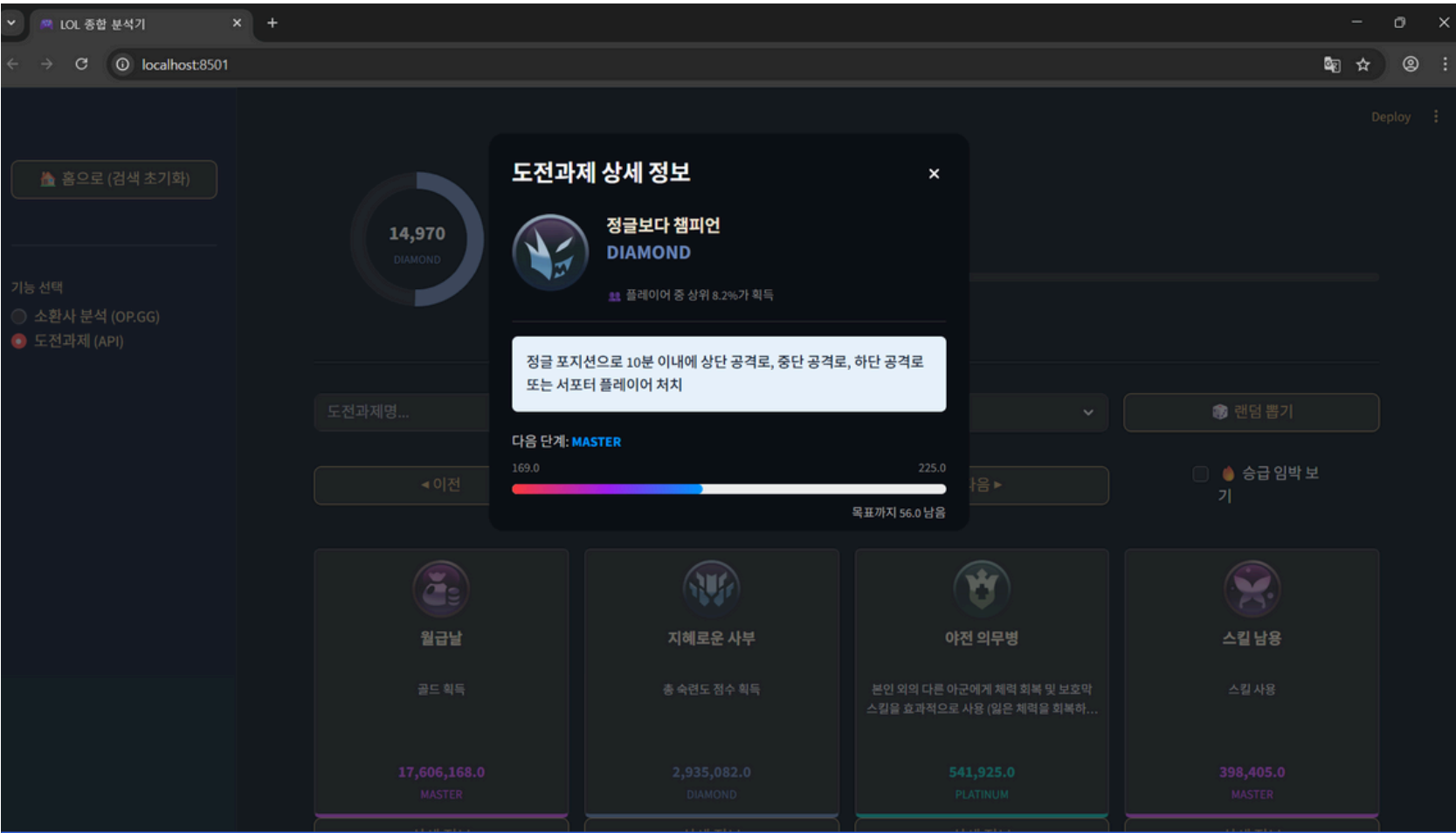
개별 검색

원하는 도전과제를 빠르게 찾는 기능



승급 임박 도전과제

이 항목을 클릭하면 개별 도전과제 표시항목 최상단에
도전과제의 다음 단계까지 얼마 남지 않은 항목이 표시됨



랜덤 뽑기

랜덤으로 한개의 도전과제의 상세 정보를 표시함

```
def make_donut(val, max_val, tier):
    per = (val/max_val*100) if max_val>0 else 0
    color = get_tier_color(tier)

    fig = go.Figure(data=[go.Pie(
        labels=['A','B'],
        values=[per, 100-per],
        hole=0.75,
        marker=dict(colors=[color, 'rgba(255,255,255,0.1)']),
        textinfo='none',
        hoverinfo='none',
        sort=False
    )])

    fig.update_layout(
        annotations=[
            dict(text=f"{val:},%", x=0.5, y=0.55, font_size=20,
                font_color="#fff", showarrow=False, font_weight="bold"),
            dict(text=tier, x=0.5, y=0.35, font_size=12,
                font_color=color, showarrow=False)
        ],
        margin=dict(l=0,r=0,t=0,b=0),
        height=140,
        paper_bgcolor='rgba(0,0,0,0)',
        plot_bgcolor='rgba(0,0,0,0)',
        showlegend=False
    )
    return fig
```

```
def get_active_game(puuid):
    if not API_KEY: return {'error': 'API_KEY_MISSING'}
    try:
        url = f"https://kr.api.riotgames.com/lol/spectator/v5/active-games/by-summoner/{puuid}"
        res = requests.get(url, headers=HEADERS_API)

        if res.status_code == 200: return res.json()
        elif res.status_code == 404: return {'error': 'GAME_NOT_FOUND'}
        elif res.status_code == 401: return {'error': 'API_ERROR_401'}
        else: return {'error': f'API_ERROR_{res.status_code}'}

    except requests.exceptions.RequestException:
        return {'error': 'NETWORK_ERROR'}
    except Exception:
        return {'error': 'UNKNOWN_ERROR'}

def change_player(new_riot_id):
    st.session_state.target_riot_id = new_riot_id
```

```
def fetch_opgg_data(name, tag):
    encoded = f"{quote(name)}-{quote(tag)}"
    try:
        r_champ = requests.get(
            f"https://op.gg/ko/lol/summoners/kr/{encoded}/champions",
            headers=HEADERS_SCRAP)
        r_mastery = requests.get(
            f"https://op.gg/ko/lol/summoners/kr/{encoded}/mastery",
            headers=HEADERS_SCRAP)
        return r_champ.text, r_mastery.text
    except: return None, None

def parse_champs(html):
    if not html: return []
    soup = BeautifulSoup(html, "html.parser")
    result = []
    for r in soup.select("tbody tr"):
        try:
            txt = r.get_text(" ", strip=True).lower()
            if "vs" in txt: continue
            img = r.find("img")
            if not img: continue
            w = re.search(r"(\d+)\s*승", txt)
            l = re.search(r"(\d+)\s*패", txt)
            wins = int(w.group(1)) if w else 0
            losses = int(l.group(1)) if l else 0
            if wins+losses == 0: continue
            result.append({"name": img["alt"], "img": img["src"],
                           "wins": wins, "losses": losses})

            if len(result) == 9: break
        except: continue
```

```
st.markdown("""
<style>
.stApp {
    background-color: #010a13;
    color: #c8aa6e;
}
.imminent-card {
    border: 2px solid #d13639 !important;
    background-color: #2a1e1e !important;
    box-shadow: 0 0 10px rgba(209, 54, 57, 0.2);
}
div.stButton > button {
    background-color: #1e2328;
    color: #c8aa6e;
    border: 1px solid #c8aa6e;
}
</style>
""", unsafe_allow_html=True)
```

PLOTY 사용

승급 임박 계산, 그리고 Plotly를 이용하여 시각화 (도넛 모양 그래프)

크롤링

fetch_opgg_data와 parse_champs 함수를 사용하여 특정 소환사의 최근 전적을 가져와 분석하였다.




API 사용

Riot API를 사용하여 실시간 게임 정보 도전과제 데이터를 가져오기

CSS 사용


Streamlit 컴포넌트의 기본 스타일을 오버라이드하여 LOL 테마의 다크 모드 UI를 구현



| | | |
|---|--|---|
|  27 판테온 271,075 |  26 요네 251,649 |  17 크산테 158,996 |
|---|--|---|

| 블루팀 (Blue Team) | |
|---|---------------------|
|  | 코르케 - YRN[H#KR1 |
|  | 밀리아 - MARCAR [H#KR1 |
|  | 울러프 - 딜플레이#KR1 |
|  | 트리스타나 - 난나의ATM#KR1 |
|  | 룰루 - 안보이달안해#KR1 |

lasund72 #7227 (검색 대상)

>  인게임 정보 (실시간 게임 확인)

1~2주차

opgg크롤링 기능 구현

streamlit을 이용한
기본적인 시각화

3~4주차

riot api를 이용해
도전과제 정보 추출

도전과제 데이터 시각화

5~6주차

css를 이용한 인터페이스
제작 및 요소 배치

api를 이용한 인게임 정보
및 부가 기능 추가

두 프로그램 병합

황보현준

opgg사이트 크롤링

streamlit을 이용한
데이터 시각화

프레젠테이션 자료 제작

공동 작업 내용

프로그램 두개 합병

부가 기능 제작

인터페이스 적용
및 시작 화면 제작

이주현

도전과제 api 사용

실시간 api를 이용한
인게임 정보 제공

인터페이스 제작

결과

riot api와 크롤링을 이용하여
게임 정보를 가져와 GUI를
통해 구현

코드를 맞춤제작으로
구현을해서 보다 보기쉽고
게임을 할때 도움이 됨

개선점

인터페이스 적용
및 시작 화면 제작

다른 사람이 사용할 수
있게 제작

api코드 갱신 필요

감사합니다