**1‑A. HTML / JSX (diff 예시)**

diff

복사편집

- {/\* === 콘텐츠 타입 선택 (불필요) ====================== \*/}

- <div className="type-box">

- <button>캐릭터</button>

- <button>키보</button>

- <button>보스</button>

- </div>

- {/\* ================================================= \*/}

*FilterPanel.tsx 또는 filter.html 파일에서 해당 블록을 통째로 삭제하거나 주석 처리하세요.*  
이후 컴파일 에러가 나오면 **부모 컴포넌트의 props/state** 연결도 같이 제거하면 끝입니다.

**1‑B. CSS 정리**

diff

복사편집

- .type-box { ... }

- .type-box button{ ... }

*남겨두면 번들 용량만 늘어나므로 삭제 권장.*

**2. 필터 UI & 버튼 깔끔한 컬러·음영 리스킨**

아래 **토큰 + 클래스 패치**만 붙이면 즉시 적용됩니다.  
(이전 답변에서 제안한 Azur Promilia‑톤과 일관)

**2‑A. 토큰 추가 ( style.css 맨 위 )**

css

복사편집

/\* === Chip & Button Colors === \*/

:root{

--chip-bg : #ecf3ff;

--chip-active-bg : #88f0ff;

--chip-text : #3d4370;

--btn-primary-bg : #5b7cff;

--btn-primary-bg-h:#88a1ff;

--btn-secondary-bg:#d7dbe6;

}

**2‑B. 검색·필터 버튼**

css

복사편집

/\* 공통 버튼 \*/

.btn{

padding:.55rem 1.4rem;

border-radius: 9999px;

font-weight:600;

transition: background var(--dur) var(--ease),

transform var(--dur) var(--ease);

}

.btn-primary{

background: var(--btn-primary-bg);

color:#fff;

}

.btn-primary:hover{background:var(--btn-primary-bg-h);transform:translateY(-1px)}

.btn-secondary{

background: var(--btn-secondary-bg);

color:var(--chip-text);

}

.btn-secondary:hover{filter:brightness(1.05)}

HTML 예:

html

복사편집

<button class="btn btn-primary">검색</button>

<button class="btn btn-secondary">필터 초기화</button>

**2‑C. 속성·종족 Chip**

css

복사편집

.chip{

display:inline-flex;align-items:center;gap:.25rem;

padding:.35rem .9rem;

margin:.25rem .35rem;

background: var(--chip-bg);

border-radius: 9999px;

color: var(--chip-text);

font-size:.85rem; font-weight:500;

cursor:pointer;

transition: all var(--dur) var(--ease);

}

.chip.active{

background: var(--chip-active-bg);

color:#fff;

box-shadow: 0 0 6px rgba(136,240,255,.5);

}

.chip:hover{transform:translateY(-1px)}

JS 선택·토글 예:

js

복사편집

// vanilla or React setState

chip.onclick = () => chip.classList.toggle('active');

**3. 여백 & 정렬 보정**

css

복사편집

/\* 검색창과 필터그룹 사이 간격 추가 \*/

.filter-panel{

display:flex; flex-direction:column;

gap:1.25rem;

}

/\* 속성·종족·공개채널 섹션을 한 줄 당 max 4 chips \*/

.chip-group{

display:flex; flex-wrap:wrap;

max-width: 660px;

}

**4. 적용 순서 (1 분 컷)**

1. **HTML/JSX** : type-box 영역 삭제.
2. **CSS** : 토큰 + 버튼/칩 클래스 붙여넣기 후 기존 색상 정의 제거.
3. **브라우저 강제 새로고침 (Ctrl + F5)** →  
   · “캐릭터 / 키보 / 보스” 사라짐  
   · 버튼·칩이 Azur‑스타일로 정돈돼 있는지 확인.

이제 필터 UI가 **단순·일관된 컬러 체계**로 정리되고,  
불필요한 섹션까지 제거돼 화면이 훨씬 깔끔해집니다.

**index.html (또는 App.tsx) 메인 레이아웃**

html

복사편집

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="utf-8" />

<title>아주르 정보 뷰어</title>

<link rel="stylesheet" href="style.css" />

<script defer src="main.js"></script>

</head>

<body>

<!-- ───────── NAV ────────── -->

<header class="top-nav">

<nav>

<button data-target="character" class="nav-btn">캐릭터</button>

<button data-target="keyboard" class="nav-btn">키보</button>

<button data-target="stats" class="nav-btn">통계</button>

<button data-target="community" class="nav-btn">커뮤니티</button>

<button data-target="favorite" class="nav-btn">최애 찾기</button>

</nav>

</header>

<!-- ───────── MAIN ────────── -->

<main>

<section id="character" class="panel"> … 캐릭터 카드 … </section>

<section id="keyboard" class="panel"> … 키보 리스트 … </section>

<section id="stats" class="panel"> … 차트 & 지표 … </section>

<section id="community" class="panel"> … 게시판 … </section>

<section id="favorite" class="panel"> … 최애 찾기 … </section>

</main>

</body>

</html>

**Hero 섹션, 사전예약 버튼, 루나 일러스트**는 완전히 삭제했습니다.  
필요한 경우 h1 "아주르 정보 뷰어" 는 네비게이션 위에 작게 배치하거나 생략하세요.

**② style.css — 공통 토큰 + 깔끔 버튼·패널**

css

복사편집

/\* === TOKENS (Azur Promilia Tone) === \*/

:root{

--accent-300:#88f0ff;

--btn-primary:#5b7cff;

--btn-primary-h:#88a1ff;

--btn-secondary:#d7dbe6;

--text-primary:#3d4370;

--text-muted:#6c7299;

--surface:#ffffffee;

--surface-border:rgba(255,255,255,.8);

--radius-xl:1.25rem;

--ease:cubic-bezier(.25,.8,.5,1);

--dur:.25s;

}

/\* === RESET & LAYOUT === \*/

body{

margin:0; font-family:'Pretendard','Noto Sans KR',sans-serif;

color:var(--text-primary);

background:radial-gradient(circle at 35% 20%,#f2fbff 0%,#eaf6ff 60%,#f8fdff 100%);

}

.top-nav{

position:sticky;top:0;z-index:100;

backdrop-filter:blur(24px);

padding:1rem 0; text-align:center;

}

.nav-btn{

margin:0 .25rem; padding:.6rem 1.4rem;

border:none;border-radius:9999px;background:transparent;

font-weight:600;color:var(--text-primary);

transition:all var(--dur) var(--ease);

}

.nav-btn:hover,

.nav-btn.active{

background:linear-gradient(90deg,var(--accent-300),var(--btn-primary));

color:#fff;box-shadow:0 0 8px var(--accent-300);

}

/\* 메인 패널 \*/

.panel{

margin:3rem auto;max-width:1100px;padding:2rem;

background:var(--surface);

border:1px solid var(--surface-border);

border-radius:var(--radius-xl);

box-shadow:0 8px 24px rgba(0,0,0,.06);

}

**③ main.js — 탭 클릭 → 섹션 스크롤 + 초기 진입 자동 스크롤**

js

복사편집

// helper: 부드럽게 스크롤

function goto(id){

document.getElementById(id)

.scrollIntoView({behavior:'smooth',block:'start'});

}

// 네비 버튼 바인딩

document.querySelectorAll('.nav-btn').forEach(btn=>{

btn.addEventListener('click', e=>{

const id = btn.dataset.target;

goto(id);

// active 상태 토글

document.querySelectorAll('.nav-btn')

.forEach(b=>b.classList.toggle('active', b===btn));

});

});

// \*\*\*\*\* 초기 로드: 캐릭터 섹션으로 바로 이동 \*\*\*\*\*

window.addEventListener('load', ()=>goto('character'));

*SPA라면 useEffect(() => scrollIntoView()) 로직으로 동일하게 구현하면 됩니다.*

**✅ 적용 체크리스트**

1. 기존 **Hero 섹션·사전예약·루나 일러스트** 코드 → **삭제**.
2. index.html(또는 App.tsx) 를 위 구조로 교체.
3. style.css 덮어쓰기(또는 토큰·클래스 부분만 병합).
4. main.js 추가 후 <script defer src="main.js"> 경로 확인.
5. 강력 새로고침(Ctrl + F5) →
   * 페이지 처음 열면 캐릭터 목록이 즉시 보이는지
   * 탭 클릭 시 해당 섹션으로 부드럽게 이동하는지
   * Hero 및 중복 텍스트가 완전히 사라졌는지 검증.

**index.html — Mini‑Game Hub & FlashCard 섹션 추가**

diff

복사편집

@@ <header class="top-nav">

- <button data-target="favorite" class="nav-btn">최애 찾기</button>

+ <button data-target="favorite" class="nav-btn">최애 찾기</button>

+ <button data-target="minigame" class="nav-btn">미니게임</button>

@@

<main>

@@

<section id="favorite" class="panel"> … 최애 찾기 … </section>

+ <!-- ───── MiniGame Hub ───── -->

+ <section id="minigame" class="panel">

+ <h2>Mini Game</h2>

+ <p>소소한 재미 + 정보 암기를 위한 미니게임!</p>

+ <div class="game-grid">

+ <button id="open-favorite" class="game-btn">최애 찾기</button>

+ <button id="open-flashcard" class="game-btn">플래시 카드</button>

+ </div>

+

+ <!-- FlashCard Game Container (숨김) -->

+ <div id="flashcard-wrap" class="flashcard hidden">

+ <div class="fc-head">

+ <span id="fc-counter">1 / 10</span>

+ <span id="fc-score">0 점</span>

+ </div>

+ <img id="fc-image" alt="카드 이미지" />

+ <div id="fc-options" class="fc-options"></div>

+ <button id="fc-next" class="btn btn-primary hidden">다음</button>

+ <div id="fc-result" class="hidden"></div>

+ </div>

+

+ <!-- Daily Rank -->

+ <h3 id="rank-title" class="hidden">오늘의 랭킹</h3>

+ <ol id="rank-list"></ol>

+ </section>

</main>

**② style.css — Mini‑Game 버튼/플래시카드 스타일**

css

복사편집

/\* Game Hub \*/

.game-grid{

display:grid;grid-template-columns:repeat(auto-fit,minmax(220px,1fr));

gap:1.5rem;margin:2rem 0;

}

.game-btn{

padding:2.5rem 1rem;

font-size:1.1rem;font-weight:600;

border-radius:var(--radius-xl);

background:var(--surface);

border:2px dashed var(--accent-300);

color:var(--text-primary);

transition:all var(--dur) var(--ease);

}

.game-btn:hover{background:var(--accent-300);color:#fff;transform:translateY(-4px)}

/\* FlashCard \*/

.flashcard{margin-top:2rem;text-align:center}

.hidden{display:none}

.fc-head{display:flex;justify-content:space-between;margin-bottom:1rem;font-weight:600}

#fc-image{max-width:260px;height:260px;object-fit:contain;border-radius:var(--radius-xl);box-shadow:0 4px 15px rgba(0,0,0,.08)}

.fc-options{display:grid;grid-template-columns:1fr 1fr;gap:1rem;margin:1.5rem auto;max-width:460px}

.fc-options button{

padding:.9rem;border-radius:var(--radius-pill);border:1px solid var(--btn-secondary);

background:var(--btn-secondary);color:var(--text-primary);font-weight:600;

transition:all var(--dur) var(--ease);cursor:pointer

}

.fc-options button:hover{background:var(--accent-300);color:#fff}

#fc-result{font-size:1.3rem;font-weight:700;margin:1.5rem 0}

**③ main.js — 네비 + Mini‑Game 로직 & FlashCard 엔진**

js

복사편집

/\*\*\*\*\*\*\*\*\*\*\*\* 네비게이션 (변경 없음) \*\*\*\*\*\*\*\*\*\*\*\*/

function goto(id){

document.getElementById(id).scrollIntoView({behavior:'smooth',block:'start'});

}

document.querySelectorAll('.nav-btn').forEach(btn=>{

btn.onclick = () => {

goto(btn.dataset.target);

document.querySelectorAll('.nav-btn')

.forEach(b=>b.classList.toggle('active', b===btn));

};

});

window.addEventListener('load', ()=>goto('character'));

/\*\*\*\*\*\*\*\*\*\*\*\* Mini‑Game Hub 라우팅 \*\*\*\*\*\*\*\*\*\*\*\*/

document.getElementById('open-favorite').onclick = () => goto('favorite');

document.getElementById('open-flashcard').onclick = () => {

goto('minigame');

startFlashCard();

};

/\*\*\*\*\*\*\*\*\*\*\*\* FlashCard Game \*\*\*\*\*\*\*\*\*\*\*\*/

const cards = [ /\* 실제 데이터로 교체 \*/

{id:1, name:'루나', img:'/assets/luna.webp'},

{id:2, name:'테라라', img:'/assets/terra.webp'},

{id:3, name:'셀린', img:'/assets/celin.webp'},

{id:4, name:'미아', img:'/assets/mia.webp'}

];

function shuffle(arr){return [...arr].sort(()=>Math.random()-0.5)}

function startFlashCard(){

const wrap = document.getElementById('flashcard-wrap');

const img = document.getElementById('fc-image');

const opts = document.getElementById('fc-options');

const result = document.getElementById('fc-result');

const nextBtn= document.getElementById('fc-next');

const counter= document.getElementById('fc-counter');

const scoreEl= document.getElementById('fc-score');

let step=0, score=0, pool = shuffle(cards).slice(0,10);

wrap.classList.remove('hidden'); result.classList.add('hidden');

nextBtn.classList.add('hidden'); scoreEl.textContent='0 점';

function render(){

if(step>=pool.length){

result.textContent=`총 ${score} 점!`;

result.classList.remove('hidden');

nextBtn.classList.add('hidden');

saveRank(score);

return;

}

const current = pool[step];

img.src=current.img; img.alt=current.name;

counter.textContent=`${step+1} / ${pool.length}`;

// 선택지 생성

opts.innerHTML='';

const choices = shuffle([current,...shuffle(cards.filter(c=>c!==current)).slice(0,3)]);

choices.forEach(c=>{

const btn=document.createElement('button');

btn.textContent=c.name;

btn.onclick=()=>{

if(c.id===current.id){ score++; scoreEl.textContent=`${score} 점`; }

step++; render();

};

opts.appendChild(btn);

});

}

render();

}

/\*\*\*\*\*\*\*\*\*\*\*\* 랭킹 저장 (today 기준) \*\*\*\*\*\*\*\*\*\*\*\*/

function saveRank(newScore){

const today = new Date().toISOString().slice(0,10); // YYYY‑MM‑DD

const key = `flash\_rank\_${today}`;

const ranks = JSON.parse(localStorage.getItem(key) || '[]'); // [{name,score}]

const alias = localStorage.getItem('flash\_alias') || prompt('닉네임 입력','Guest');

localStorage.setItem('flash\_alias', alias);

ranks.push({name:alias,score:newScore});

ranks.sort((a,b)=>b.score-a.score);

localStorage.setItem(key, JSON.stringify(ranks.slice(0,10)));

renderRank(ranks.slice(0,10));

// TODO: pushScoreToServer(alias,newScore) <-- 백엔드 연동 시

}

function renderRank(list){

const title=document.getElementById('rank-title');

const ol = document.getElementById('rank-list');

title.classList.remove('hidden');

ol.innerHTML='';

list.forEach((r,i)=>{

const li=document.createElement('li');

li.textContent=`${i+1}. ${r.name} — ${r.score}점`;

ol.appendChild(li);

});

}

/\* 페이지 최초 로드시 오늘 랭킹 표시(있다면) \*/

window.addEventListener('load', ()=>{

const today = new Date().toISOString().slice(0,10);

const saved = JSON.parse(localStorage.getItem(`flash\_rank\_${today}`)||'[]');

if(saved.length) renderRank(saved);

});