RA 게임 홈페이지 API 연동 명세서

📋 목차

1. 인증 API
2. 사용자 API
3. 소식 API
4. 커뮤니티 API
5. 게임 정보 API
6. 다운로드 API

🔐 인증 API

1. 로그인 (login.html)

Endpoint: `POST /api/auth/login`

Request Body:

```json
{
  "email": "user@example.com",
  "password": "password123",
  "remember": true
}
```

Response (200 OK):

```json
```

```
{
  "success": true,
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs...",
    "user": {
      "id": "user_123",
      "username": "플레이어123",
      "email": "user@example.com",
      "avatar": "https://cdn.example.com/avatars/123.png"
    }
  },
  "message": "로그인 성공"
}
```


프론트엔드 처리:

```javascript
```

```javascript
// login.html에 추가할 코드
document.getElementById('loginForm').addEventListener('submit', async (e) => {
    e.preventDefault();

    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const remember = document.getElementById('remember').checked;

    try {
        const response = await fetch('/api/auth/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ email, password, remember })
        });

        const result = await response.json();

        if (result.success) {
            // 토큰 저장
            localStorage.setItem('accessToken', result.data.accessToken);
            localStorage.setItem('user', JSON.stringify(result.data.user));

            // 메인 페이지로 이동
            window.location.href = 'index.html';
        } else {
            alert(result.message);
        }
    } catch (error) {
        console.error('로그인 오류:', error);
        alert('로그인 중 오류가 발생했습니다.');
    }
});
```

---

2. 회원가입 (signup.html)

Endpoint: `POST /api/auth/signup`

Request Body:

```json
```

```
{
  "username": "플레이어123",
  "email": "user@example.com",
  "password": "SecurePass123!",
  "birthdate": "1990-01-01",
  "region": "kr",
  "marketing": true
}
```

Response (201 Created):

```
json

{
  "success": true,
  "data": {
    "userId": "user_123",
    "username": "플레이어123",
    "email": "user@example.com"
  },
  "message": "회원가입이 완료되었습니다. 이메일을 확인해주세요."
}
```

---

3. 소셜 로그인

Endpoint: `GET /api/auth/social/{provider}`

- provider: google, facebook, discord

Response: OAuth 리다이렉트

👤 사용자 API

1. 사용자 정보 조회

Endpoint: `GET /api/user/profile`

Headers:

```
Authorization: Bearer {accessToken}
```

Response (200 OK):

```
json
```

```
{
  "success": true,
  "data": {
    "id": "user_123",
    "username": "플레이어123",
    "email": "user@example.com",
    "level": 75,
    "class": "워리어",
    "guildName": "이터널",
    "joinDate": "2025-01-15T00:00:00Z",
    "avatar": "https://cdn.example.com/avatars/123.png"
  }
}
```

📰 소식 API

1. 소식 목록 조회 (news.html)

Endpoint: `GET /api/news/list`

Query Parameters:

- page: 페이지 번호 (default: 1)
- limit: 페이지당 개수 (default: 10)
- category: 카테고리 필터 (전체, 업데이트, 이벤트, 공지사항, 점검)

Request:

```
GET /api/news/list?page=1&limit=10&category=업데이트
```

Response (200 OK):

```
json
```

```
{
  "success": true,
  "data": {
    "news": [
      {
        "id": "news_001",
        "title": "v2.5.0 신규 업데이트 출시",
        "category": "업데이트",
        "excerpt": "새로운 레이드 던전 '고대의 신전'이 추가되었습니다...",
        "thumbnail": "https://cdn.example.com/news/001.jpg",
        "author": "RA Team",
        "date": "2025-11-26T10:00:00Z",
        "views": 1248,
        "featured": false
      }
    ],
    "pagination": {
      "currentPage": 1,
      "totalPages": 5,
      "totalItems": 48,
      "hasNext": true,
      "hasPrev": false
    }
  }
}
```


프론트엔드 처리:

```
javascript
```

```
// news.html에 추가할 코드
async function loadNews(page = 1, category = '전체') {
    try {
        const response = await fetch(
            `/api/news/list?page=${page}&limit=10&category=${category}`
        );
        const result = await response.json();

        if (result.success) {
            displayNews(result.data.news);
            displayPagination(result.data.pagination);
        }
    } catch (error) {
        console.error('소식 로딩 오류:', error);
    }
}

function displayNews(newsList) {
    const newsContainer = document.querySelector('.news-list');
    newsContainer.innerHTML = newsList.map(news => `
        <article class="news-item">
            <div class="news-thumbnail">
                <img src="${news.thumbnail}" alt="${news.title}">
            </div>
            <div class="news-details">
                <div class="news-meta">
                    <span class="news-category">${news.category}</span>
                    <span class="news-date">📅 ${formatDate(news.date)}</span>
                </div>
                <h3>${news.title}</h3>
                <p class="news-excerpt">${news.excerpt}</p>
                <a href="news-detail.html?id=${news.id}" class="read-more">
                    자세히 보기 →
                </a>
            </div>
        </article>
    `).join('');
}
```

---

2. 소식 상세 조회

Endpoint: `GET /api/news/{id}`

Response (200 OK):

```json
{
  "success": true,
  "data": {
    "id": "news_001",
    "title": "v2.5.0 신규 업데이트 출시",
    "category": "업데이트",
    "content": "<p>상세 내용 HTML...</p>",
    "thumbnail": "https://cdn.example.com/news/001.jpg",
    "author": "RA Team",
    "date": "2025-11-26T10:00:00Z",
    "views": 1248,
    "tags": ["업데이트", "레이드", "PVP"]
  }
}
```

💬 커뮤니티 API

1. 게시글 목록 조회 (community.html)

Endpoint: `GET /api/community/posts`

Query Parameters:

- page: 페이지 번호
- limit: 페이지당 개수
- category: 카테고리 (전체, 자유, 질문/답변, 공략/팁 등)
- sort: 정렬 (latest, popular, views)

Request:

```
GET /api/community/posts?page=1&limit=20&category=공략/팁&sort=popular
```

Response (200 OK):

```json
```

```
{
  "success": true,
  "data": {
    "posts": [
      {
        "id": "post_001",
        "title": "신규 레이드 공략법 완벽 정리",
        "category": "공략/팁",
        "author": {
          "id": "user_456",
          "username": "레이드마스터",
          "avatar": "https://cdn.example.com/avatars/456.png"
        },
        "excerpt": "고대의 신전 레이드 1~3페이즈까지 완벽 공략법...",
        "tags": ["공략", "레이드"],
        "createdAt": "2025-11-27T12:00:00Z",
        "stats": {
          "views": 1248,
          "likes": 156,
          "comments": 43
        },
        "isHot": true
      }
    ],
    "pagination": {
      "currentPage": 1,
      "totalPages": 10,
      "totalItems": 195
    }
  }
}
```


프론트엔드 처리:

```
javascript
```

```
// community.html에 추가할 코드
async function loadPosts(category = '전체', page = 1) {
    try {
        const response = await fetch(
            `/api/community/posts?page=${page}&limit=20&category=${category}`
        );
        const result = await response.json();

        if (result.success) {
            displayPosts(result.data.posts);
            displayPagination(result.data.pagination);
        }
    } catch (error) {
        console.error('게시글 로딩 오류:', error);
    }
}
```

---

2. 게시글 작성

Endpoint: `POST /api/community/posts`

Headers:

```
Authorization: Bearer {accessToken}
```

Request Body:

```
json

{
  "title": "게시글 제목",
  "content": "게시글 내용...",
  "category": "공략/팁",
  "tags": ["공략", "레이드"]
}
```

---

3. 게시글 검색

Endpoint: `GET /api/community/search`

Query Parameters:

- q: 검색어
- page: 페이지 번호

🎮 게임 정보 API

1. 게임 정보 조회 (game-info.html)

Endpoint: `GET /api/game/info`

Response (200 OK):

json


```
{
  "success": true,
  "data": {
    "features": [
      {
        "id": "feature_001",
        "icon": "⚔️",
        "title": "실시간 액션 전투",
        "description": "다이나믹한 콤보 시스템..."
      }
    ],
    "classes": [
      {
        "id": "class_warrior",
        "name": "워리어",
        "icon": "⚔️",
        "description": "근거리 전투의 달인...",
        "stats": {
          "attack": 9,
          "defense": 8,
          "speed": 6
        }
      }
    ],
    "stats": {
      "totalDownloads": 5000000,
      "activeUsers": 1000000,
      "averageRating": 4.8,
      "supportedCountries": 50
    }
  }
}
```

📥 다운로드 API

1. 다운로드 링크 조회 (download.html)

Endpoint: `GET /api/download/links`

Response (200 OK):

```json
```

```
{
  "success": true,
  "data": {
    "platforms": [
      {
        "platform": "windows",
        "version": "2.5.0",
        "fileSize": "15.2 GB",
        "downloadUrl": "https://download.example.com/ra-windows-2.5.0.exe",
        "releaseDate": "2025-11-26T00:00:00Z",
        "checksum": "sha256:abc123..."
      },
      {
        "platform": "macos",
        "version": "2.5.0",
        "fileSize": "14.8 GB",
        "downloadUrl": "https://download.example.com/ra-macos-2.5.0.dmg",
        "releaseDate": "2025-11-26T00:00:00Z",
        "checksum": "sha256:def456..."
      }
    ],
    "systemRequirements": {
      "windows": {
        "minimum": {
          "os": "Windows 10 64-bit",
          "processor": "Intel Core i5-4460",
          "memory": "8 GB RAM",
          "graphics": "NVIDIA GTX 960",
          "storage": "20 GB"
        },
        "recommended": {
          "os": "Windows 11 64-bit",
          "processor": "Intel Core i7-8700K",
          "memory": "16 GB RAM",
          "graphics": "NVIDIA RTX 3060",
          "storage": "SSD 20 GB"
        }
      }
    }
  }
}
```

🔧 공통 설정

API Base URL

- Development: `http://localhost:3000/api`
- Production: `https://api.ra-game.com/api`

인증 헤더

모든 보호된 엔드포인트는 다음 헤더 필요:

```
Authorization: Bearer {accessToken}
```

에러 응답 형식

```json
{
  "success": false,
  "error": {
    "code": "AUTH_FAILED",
    "message": "인증에 실패했습니다.",
    "details": "토큰이 만료되었습니다."
  }
}
```

공통 에러 코드

- `400` - Bad Request (잘못된 요청)
- `401` - Unauthorized (인증 필요)
- `403` - Forbidden (권한 없음)
- `404` - Not Found (리소스 없음)
- `429` - Too Many Requests (요청 제한 초과)
- `500` - Internal Server Error (서버 오류)

📦 프론트엔드에서 사용할 유틸리티

API 클라이언트 (api.js)

```javascript
```

```
// 모든 HTML 파일에서 사용할 공통 API 클라이언트
class APIClient {
    constructor(baseURL) {
        this.baseURL = baseURL;
    }

    async request(endpoint, options = {}) {
        const token = localStorage.getItem('accessToken');

        const config = {
            ...options,
            headers: {
                'Content-Type': 'application/json',
                ...(token && { 'Authorization': `Bearer ${token}` }),
                ...options.headers
            }
        };

        try {
            const response = await fetch(`${this.baseURL}${endpoint}`, config);
            const data = await response.json();

            if (!response.ok) {
                throw new Error(data.error?.message || '요청 실패');
            }

            return data;
        } catch (error) {
            console.error('API 요청 오류:', error);
            throw error;
        }
    }

    get(endpoint) {
        return this.request(endpoint, { method: 'GET' });
    }

    post(endpoint, body) {
        return this.request(endpoint, {
            method: 'POST',
            body: JSON.stringify(body)
        });
    }

    put(endpoint, body) {
        return this.request(endpoint, {
```

```
            method: 'PUT',
            body: JSON.stringify(body)
        });
    }

    delete(endpoint) {
        return this.request(endpoint, { method: 'DELETE' });
    }
}

// 사용 예시
const api = new APIClient('http://localhost:3000/api');

// 로그인
await api.post('/auth/login', { email, password });

// 뉴스 조회
await api.get('/news/list?page=1');

// 게시글 작성
await api.post('/community/posts', { title, content });
```

🚀 다음 단계

1. **백엔드 개발 착수**
   - Node.js/Express 또는 Spring Boot 선택
   - 위 API 명세에 따라 엔드포인트 구현
2. **데이터베이스 스키마 설계**
   - Users, News, Posts 등 테이블 설계
3. **프론트엔드에 API 연동**
   - 각 HTML 파일에 API 호출 로직 추가
   - api.js 유틸리티 통합
4. **인증/인가 구현**
   - JWT 토큰 발급 및 검증
   - 보호된 라우트 설정