# SICHIMI 메타데이터 등록 포털 개발 사양서 (개발자용)

## 1. 목적

SICHIMI 에 참여하는 대학이 제공하는 메타데이터(XML 파일 또는 URL)를 등록할 수 있는 웹 기반 포털을 개발합니다. 이 포털은 유효성 검사를 자동으로 수행하고, 승인된 메타데이터만 Metadata Aggregator 에 자동 반영합니다.

## 2. 주요 기능 요약

| 기능 | 설명 |
|---|---|
| **관리자 로그인** | OIDC 기반 인증 (Keycloak 연동) |
| **메타데이터 제출** | 파일 업로드 또는 URL 입력 방식 |
| **유효성 검사 자동화** | 등록 즉시 SAML 메타데이터 스키마 및 규칙 기반 검사 수행 |
| **승인 프로세스** | 관리자 승인 후 aggregator 디렉터리로 반영 |
| **메타데이터 목록** | 등록 현황 목록 (상태: 승인 / 대기 / 오류 / 만료예정 등) |
| **이력 및 알림** | 등록/수정 이력 관리 및 유효기간 알림 기능 |

## 3. 화면 UI 구성

| 구분 | 내용 |
|---|---|
| **로그인 화면** | ▪ 로그인 버튼 (Keycloak OIDC 기반) |
| **메타데이터 등록 화면** | ▪ 업로드 방식 선택: [파일 업로드] or [URL 입력]<br>▪ 제출 버튼 → 유효성 검사 → 결과 메시지 출력 |
| **메타데이터 관리 목록** | ▪ 항목: 기관명, EntityID, 등록일, 만료일, 상태, 액션(보기, 승인)<br>▪ 필터링: 상태별, 기간별 |
| **승인 상세 화면 (운영자 전용)** | ▪ 메타데이터 원본 보기 (XML 뷰)<br>▪ 유효성 검사 로그 확인<br>▪ [승인] [반려] 버튼 |

## 4. 유효성 검사 로직 (자동 실행)

| 순서 | 내용 |
|---|---|
| ① 스키마(XSD) 검증 | xmllint --noout --schema saml-schema-metadata-2.0.xsd metadata.xml |
| ② EntityID 중복 확인 | 등록된 EntityID 목록(DB 또는 파일)과 비교 |
| ③ URI 형식 및 접근 가능성 확인 | URL 형식 유효성, HTTP 응답 여부 체크 |
| ④ 필수 항목 존재 여부 | EntityDescriptor, IDPSSODescriptor 또는 SPSSODescriptor, ContactPerson, Organization |
| ⑤ validUntil 속성 체크 | 유효기간 초과 여부 판단 |
| ⑥ 중복 RoleDescriptor 탐지 | - |
| ⑦ 전자서명(Signature) 검증 | A. xmlsec1 --verify metadata.xml |

※ 검사 결과는 JSON 형태로 반환하며, UI 상에 오류 항목과 메시지를 표시합니다.

## 5. 샘플 백엔드 코드 (Python Flask 예시)

※ 위 코드는 xmllint 를 활용한 유효성 검사의 예시이며, 실제 구현 시 XML 파싱 및 로깅, 인증, 승인 처리를 추가해야 합니다.

```
from flask import Flask, request, jsonify
import subprocess

app = Flask(__name__)

@app.route("/validate", methods=["POST"])
def validate_metadata():
    uploaded_file = request.files['metadata']
    path = f"/tmp/{uploaded_file.filename}"
    uploaded_file.save(path)

    cmd = ["xmllint", "--noout", "--schema", "/schemas/saml-schema-metadata-
2.0.xsd", path]
    result = subprocess.run(cmd, capture_output=True, text=True)

    if result.returncode == 0:
        return jsonify({"status": "valid"})
    else:
        return jsonify({"status": "invalid", "error": result.stderr}), 400

if __name__ == "__main__":
    app.run(debug=True)
```

## 6. 권장 스택

| 구분 | 스택 |
|---|---|
| 프론트엔드 | React 또는 Vue.js |
| 백엔드 | Python(Flask, FastAPI), 또는 Node.js 기반 API |
| 인증 | Keycloak 연동 OIDC 로그인 |
| 스토리지 | 파일시스템 + DB (등록 정보 + 상태 관리) |
| 배포 | Docker 컨테이너 + Nginx 프록시 |