SW사관학교 JUNGLE TEAM3

<Session Volume Control Panel>

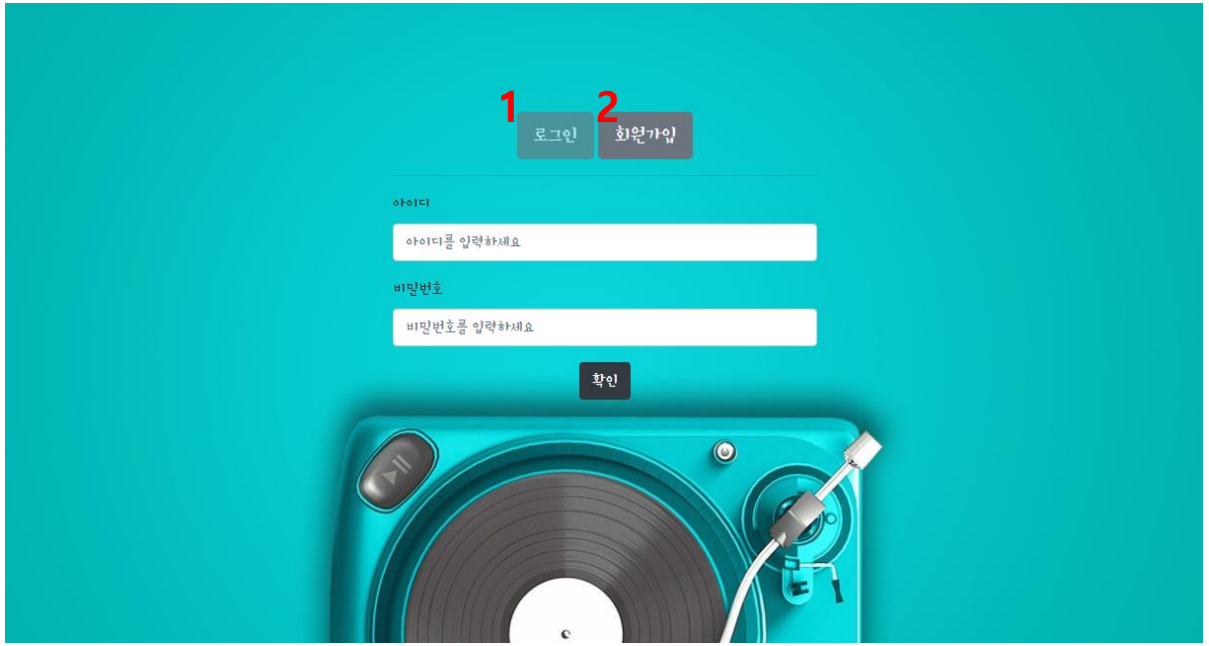
김승현, 이병호, 조새힘

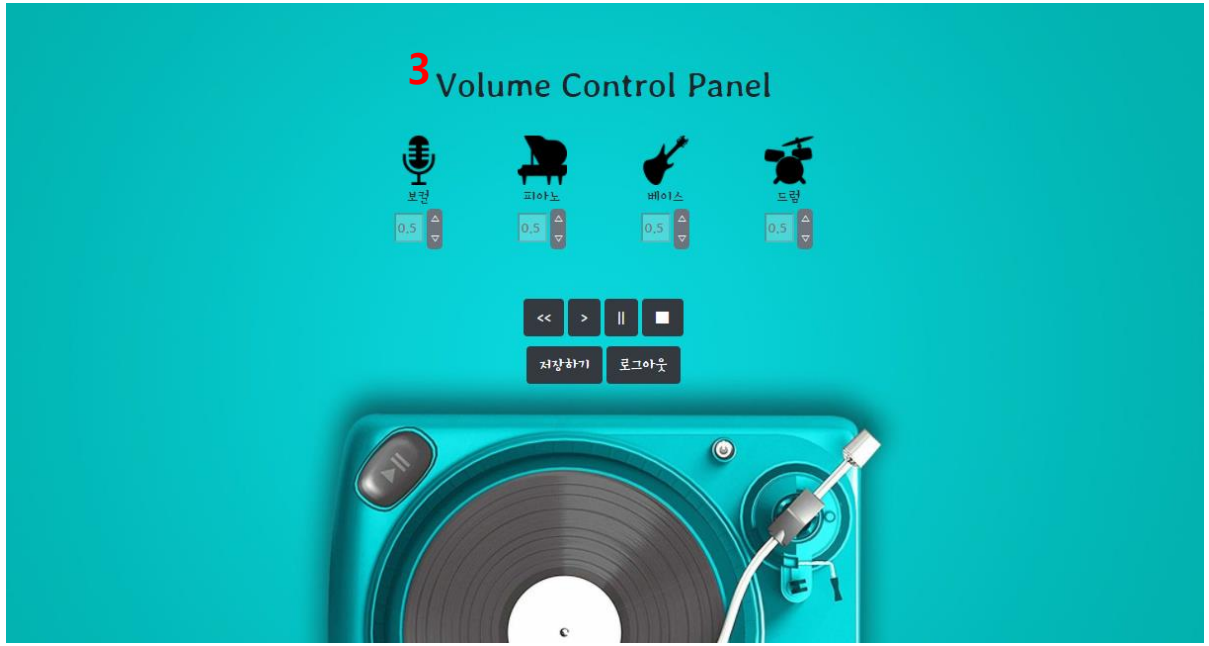
목차

I. 기획 의도

- ✓ 프로젝트 설명
  - ✓ motive: 믹싱 콘솔 프로그램
  - ✓ 각 세션의 밸런스를 조절하여 원하는 음악 만들기!

- ✓ 페이지 구성





II. 로그인 기능

- ✓ 쿠키/세션이란?
  - ✓ Html의 비연결성을 보완하기 위해 등장
  - ✓ 사용자 연결을 유지하기 위해 서버에서 상태를 저장

- ✓ JWT 인증 방식?
  - ✓ 토큰을 이용해 인증, JWT는 웹에서 사용할 수 있는 엑세스 토큰을 다루는 표준
  - ✓ 장점 : 세션과 다르게 서버에 부하를 주지 않음
  - ✓ 단점 : 토큰이 공격자에게 탈취될 수 있으므로 보안에 주의해야함

II. 로그인 기능

✓ 세션 방식의 구현

1. 로그인

```python
@app.route('/api/login', methods=['POST', 'GET'])
def login():
    if request.method == 'GET':
        return render_template("index.html")

    user_id = request.form['userId']
    user_password = request.form['password']
    if db.users.find_one({'userid': user_id,
                          'password': user_password}) is not None:
        session['userid'] = request.form['userId']
    return jsonify({'result': 'success'})
```

- Flask의 Session을 import 하여 구현
- DB에 ID가 존재할 시 Session에 해당 id 추가

2. 세션 유지

```python
@app.route('/')
def home():
    if 'userid' in session:
        return render_template("play.html")
    else:
        print('세션 off')

    return render_template("index.html")
```

- 홈으로 들어왔을 때 session이 존재한다면 바로 play.html로 이동

3. 로그아웃

```python
@app.route('/api/logout', methods=['GET'])
def logout():
    session.pop('userid', None)
    return redirect('/')
```

- 해당 id의 Session을 pop

II. 로그인 기능

✓ JWT 인증 방식의 구현

1. 로그인

```
@app.route('/api/login', methods=['POST', 'GET'])
if(user_id == user['userid']
   and check_password_hash(user['password'], password)):
 access_token = create_access_token(identity=user_id,
                                    expires_delta=False)
 refresh_token = create_refresh_token(identity=user_id,
                                    expires_delta=False)
 resp = jsonify({'result': 'success'})
 set_access_cookies(resp, access_token)
 set_refresh_cookies(resp, refresh_token)
 return resp
```

- Flask_jwt_extended 패키지를 사용하여 구현
- DB에 ID가 존재할 시 토큰 발급

3. 로그아웃

```
@app.route('/api/logout', methods=['POST'])
def logout():
    resp = jsonify({'result': 'success'})
    unset_jwt_cookies(resp)
    return resp
```

- 해당 id의 Jwt_cookie를 제거

2. 세션 유지

```
@app.route('/')
@jwt_optional
def home():
    cur_user = get_jwt_identity()
    if cur_user is not None:
        print('로그인 유저', cur_user)
        user = db.users.find_one({'userid': cur_user})
        return render_template("play.html")
    return render_template("index.html")
```

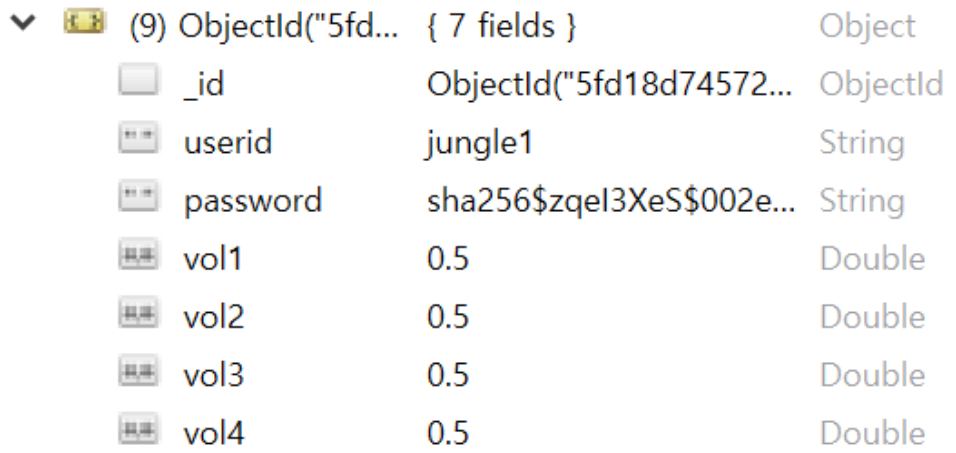
- jwt_optional 조건
- 홈으로 들어왔을 때 토큰이 존재한다면 바로 play.html로 이동

II. 로그인 기능

✓ 비밀번호 암호화

```
generate_password_hash(password, 'sha256')
```

- werkzeug.security 패키지의
  generate_password_hash를 import 하여 구현



- DB에 비밀번호가 해시 형태로 저장

✓ 회원가입 예외 처리

```
user_id = request.form['userId']
password = request.form['password']

# db에서 해당 id 찾기
user = db.users.find_one({'userid': user_id})
# 해당 아이디 이미 있으면 fail
if (user != None):
    return jsonify({'result': 'same'})
# 아이디와 비밀번호가 공백이면 fail
if (str(user_id) == '' or str(password) == ''):
    return jsonify({'result': 'empty'})
```

- 회원가입 Fail 반환

1. 이미 존재하는 아이디
2. 입력이 공백일 경우

III. Jinja2 템플릿 서버사이드 렌더링

- ✓ 기능
  - ✓ Volume control panel 페이지 로딩 시, 각 user 가 저장한 세션의 볼륨을 불러오기

- ✓ 구현
  - ✓ 처음 페이지 로딩 시, 저장된 볼륨들을 각 변수에 넣어주기

```
user = db.users.find_one({'userid': cur_user})
return render_template("play.html",
                       _vol1 = user['vol1'],
                       _vol2 = user['vol2'],
                       _vol3 = user['vol3'],
                       _vol4 = user['vol4'])
```

app.py: 페이지 렌더링 시, DB에서 해당 user의 volume 정보를 적용

```
$(document).ready(function () {
    audio1.volume = {{_vol1}};
    audio2.volume = {{_vol2}};
    audio3.volume = {{_vol3}};
    audio4.volume = {{_vol4}};
});
```

Play.html: 페이지 처음 로딩 시, DB에서 불러온 값으로 변수값 설정

IV. 개선사항

- ✓ Get 방식으로 홈이 아닌 API에 직접 접근할 수 있는 문제
  - ✓ 내부적으로만 접근해야 하는 엔드 포인트의 외부 접근을 차단
  - ✓ Ex : 서버 방화벽 이용, 웹서버 설정, remote address를 직접 차단

- ✓ 비밀번호 최소 요구 조건 추가

```
function chkPW(){

 var pw = $("#password").val();
 var num = pw.search(/[0-9]/g);
 var eng = pw.search(/[a-z]/ig);
 var spe = pw.search(/[`~!@@#$%^&*|₩₩₩'₩";:₩/?]/gi);

 if(pw.length < 8 || pw.length > 20){

  alert("8자리 ~ 20자리 이내로 입력해주세요.");
  return false;
 }else if(pw.search(/\s/) != -1){
  alert("비밀번호는 공백 없이 입력해주세요.");
  return false;
 }else if(num < 0 || eng < 0 || spe < 0 ){
  alert("영문,숫자, 특수문자를 혼합하여 입력해주세요.");
  return false;
 }else {
       console.log("통과");
    return true;
 }

}
```

- ✓ 새로 고침을 고려한 Server-side와 Customer-side rendering 사용 여부의 결정

- ✓ 사용자 사운드 파일 업로드 기능 추가

```
const elImage = document.querySelector("#reviewImageFileOpenInput");
elImage.addEventListener("change", (evt) => {
  const image = evt.target.files[0];
  if(!validImageType(image)) {
    console.warn("invalide image file type");
    return;
  }
});

function valideImageType(image) {
  const result = ([ 'image/jpeg',
                    'image/png',
                    'image/jpg' ].indexOf(image.type) > -1);
  return result;
}
```

- ✓ 서버 DB의 개인정보 암호화

구 분			암호화 기준
정보통신망, 보조저장매체를 통한 송 · 수신 시	비밀번호, 바이오정보, 고유식별정보		암호화 송 · 수신 ※ 내부망에서 전송시 해설 내용 참조
개인정보처리시스템에 저장 시	비밀번호		일방향 암호화 저장
	바이오정보		암호화 저장
	고유식별정보	인터넷 구간, 인터넷 구간과 내부망의 중간 지점(DMZ)	암호화 저장
		내부망에 저장	암호화 저장 또는 다음 항목에 따라 암호화 적용여부 · 적용범위를 정하여 저장 ① 개인정보 영향평가 대상이 되는 공공기관의 경우, 그 개인정보 영향평가의 결과 ② 위험도 분석에 따른 결과
업무용 컴퓨터, 모바일 기기에 저장시	비밀번호, 바이오정보, 고유식별정보		암호화 저장(비밀번호는 일방향 암호화 저장)

- ✓ 쿠키가 아닌 Http header를 통해서 token을 주고 session 저장소에 저장

감사합니다.