21. 선택적 도전 과제-4

1. 요구사항

- 중복 로그인 방지 기능을 추가하세요

2. 개요

- db에 있는 refresh 토큰을 새로운 토큰으로 만들어 기존 로그인을 방지하자.
- 엔티티에 access token의 정보도 추가하여, 가장 마지막으로 발급받은 access token과 db의 access token의 정보가 다르다면 토큰 인증이 안되도록 추가하자

3. 엔티티 변경

- accessToken 필드 추가하자.

```
@Column({ nullable: true })
accessToken: string;
```

4. 로그인 서비스 변경

- access 토큰 정보를 그때그때 db에 담아주도록 하자.
- 로그인 로직 변경
- refresh 토큰 뿐만 아니라 access 토큰도 담아준다.

```
async signIn(username: string, pass: string): Promise<{ access_token: string,
refresh_token: string }> {
    const user = await this.usersService.findByUsername(username);
    if (!user) {
      throw new UnauthorizedException();
    }

    if (user.loginAttempts >= 5) {
      throw new UnauthorizedException('최대 로그인 시도 횟수를 초과하였습니다.');
    }

    if (!(await this.comparePasswords(pass, user.password))) {
```

```
      user.loginAttempts += 1;
      await this.userRepository.save(user);
      throw new UnauthorizedException();
    }


    // 로그인 성공 시, 로그인 시도 횟수 초기화
    user.loginAttempts = 0;
    await this.userRepository.save(user);


    const payload = { username: user.username, sub: user.id, role: user.role };
    const refresh_token = await this.jwtService.signAsync(payload, {
      expiresIn: '7d'  // Refresh 토큰 유효기간 설정
    });
    const access_token = await this.jwtService.signAsync(payload, {
      expiresIn: '60s'
    });

    // refresh token, access token 업데이트
    user.refreshToken = refresh_token;
    user.accessToken = access_token;
    await this.userRepository.save(user);

    return {
      access_token,
      refresh_token,
    };
  }
```

5. refresh 로직 변경

- 다음과 같이 access token 정보를 db에 담아준다.

```
const user = await this.usersService.findByUsername(payload.username);
user.accessToken = newAccessToken;
```

```
async refreshAccessToken(refreshToken: string): Promise<{ access_token: string }> {

    console.log('refreshToken : ' + refreshToken);

    try {
      const payload = await this.jwtService.verifyAsync(refreshToken);

      // 'exp' 속성 제거
      delete payload.exp;

      const newAccessToken = await this.jwtService.signAsync(payload, {
        expiresIn: '60s'
      });

      const user = await this.usersService.findByUsername(payload.username);
      user.accessToken = newAccessToken;

      return { access_token: newAccessToken };
    } catch (error) {
      console.log(error);

      throw new UnauthorizedException('Refresh token is invalid');
    }
  }
```

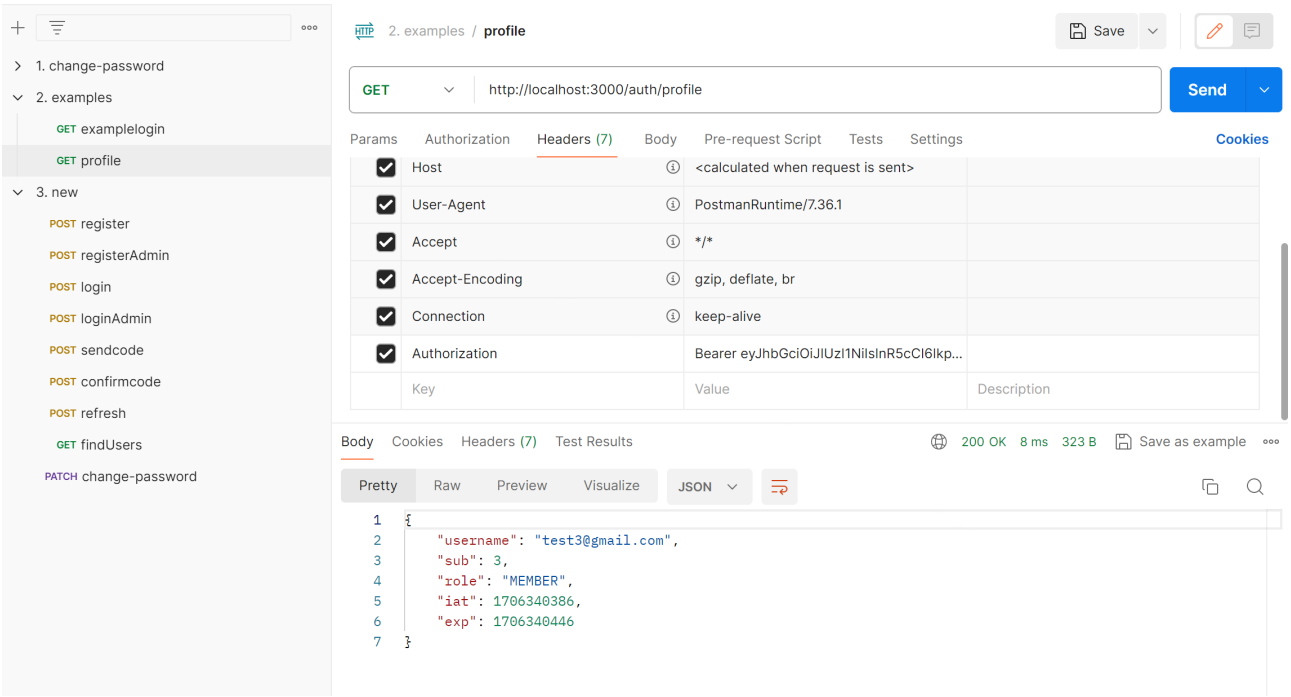
6. guard 내용 변경

- 다음과 같이 토큰에서 정보를 가져왔는데 그 토큰이 db에 저장된 토큰과 내용이 다르다면 그 어떤 인증 절차도 수행할 수 없게끔 바꾸자.

```
const user = await this.usersService.findByUsername(payload.username);
if (user.accessToken !== token) {
throw new UnauthorizedException();
}
```

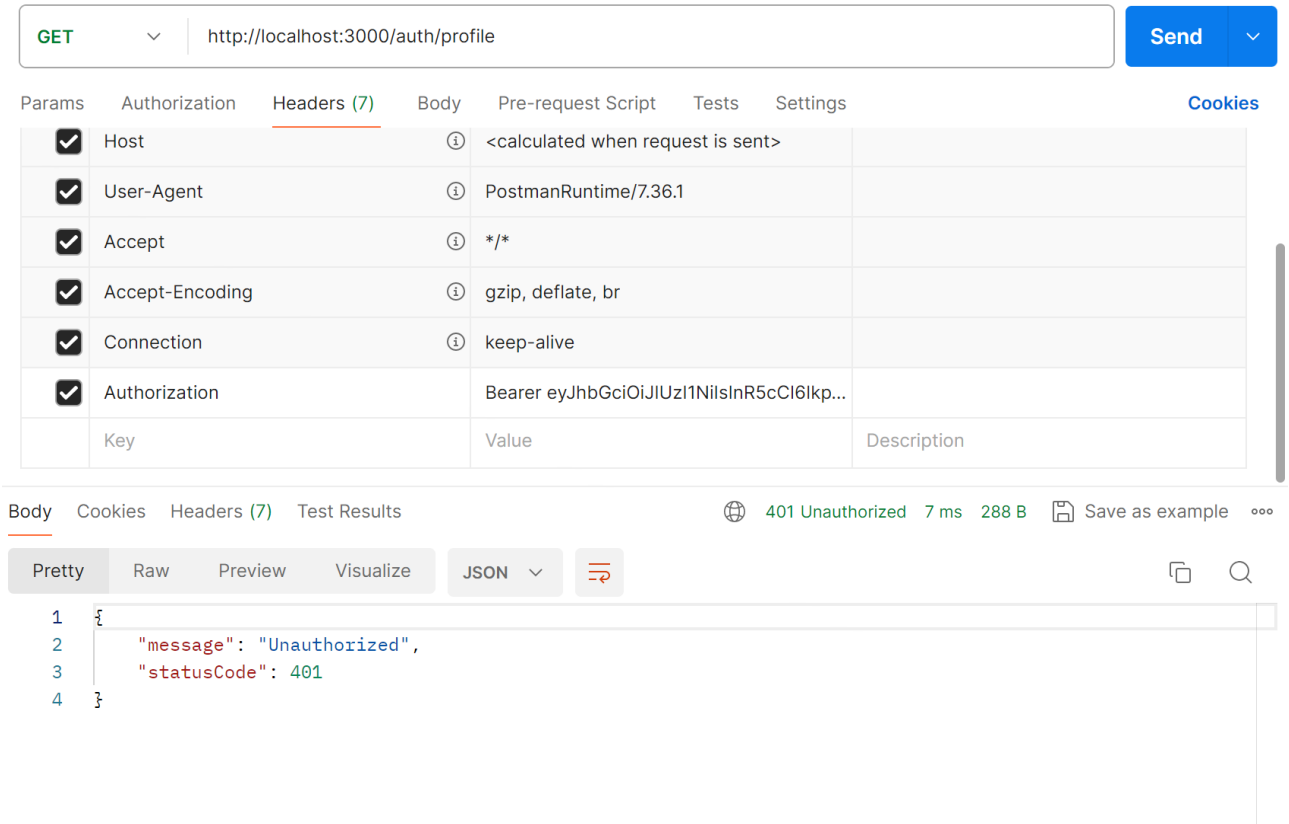
7. 확인

- 로그인 첫 시도



다음과 같이 잘 된다.

- 로그인을 하고, access token 복사 - 다시 로그인 후 기존 복사 코드로 인증 진행



- 바로 인증 안되는 것을 확인할 수 있다.