3/14 개발일지

1. 로그아웃 구현 완벽완료

▼ 로그아웃 구현을 완벽히 구현 성공했다.

- 프론트에서의 요청
  - 로컬스토리지에서 토큰을 뽑으면 web1 (일반회원)
  - 쿠키에서 토큰을 뽑으면 google (구글인증로그인회원)
  - POST형식으로, Header에 토큰저장해서 보내고, body에 {provider : web1 or google} json으로

```
const handleLogout = async (e) => {
     e.preventDefault();
     const tokenFromLocalStorage = localStorage.getI
     const tokenFromCookies = cookies.token;

     console.log("tokenFromLocalStorage:", tokenFrom
     console.log("tokenFromCookies:", tokenFromCooki
     // Determine the token source and provider
     let token, provider;
     if (tokenFromLocalStorage) {
         token = tokenFromLocalStorage;
         provider = "web1"; // Set provider for loca
     } else if (tokenFromCookies) {
         token = tokenFromCookies;
         provider = "google"; // Set provider for co
     }

     console.log(token, provider);

     if (token) {
         try {
             const response = await fetch('http://lo
                 method: 'POST',
```

```
                    headers: {
                        'Authorization': `Bearer ${toke
                        'Content-Type': 'application/js
                    },
                    body: JSON.stringify({ "provider" :
                });

                const data = await response.json();
                console.log(data);
                if (response.ok) {
                    console.log('Logout successful');
                    setIsLoggedIn(false); // Update log
                    localStorage.removeItem('token'); /
                    removeCookie('token');
                    navigate ('/'); // Remove token coo
                } else {
                    console.log('Logout failed');
                }



            } catch (error) {
                console.error('An error occurred:', err
            }
        }
    }
```

- 백엔드에서의 활동
  - 요청헤더에서 json뽑아내고 body에서 provider 뽑아냄
  - checkToken 메소드와 isExpired 메소드로 유효성과 시간 확인 (이경우엔 어차피 로그아웃이니 어떤경우든 상관없이 delete함)

```
// 로그아웃 시나리오
    @PostMapping("/member/logouts")
    public ResponseEntity<?> logout(HttpServletRequest
        System.out.println("--------로그아웃 시작--------"
```

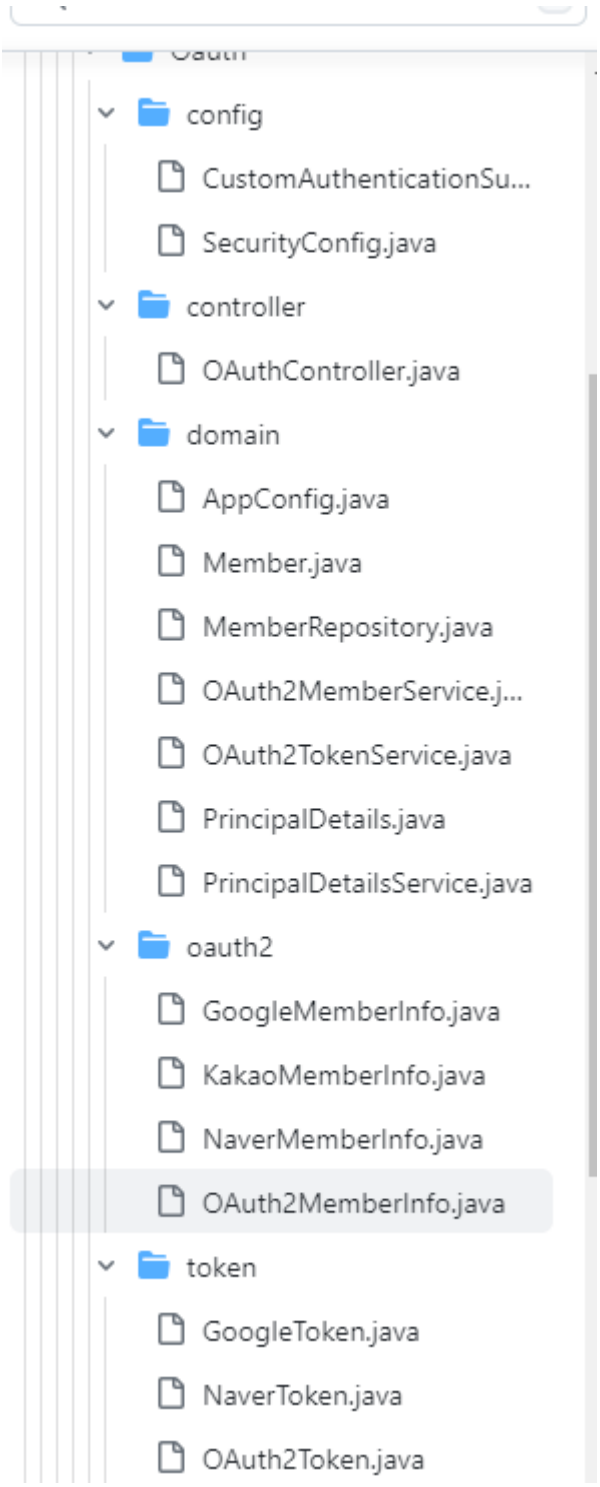
```
        // Authorization 헤더에서 토큰 값 추출
        String authorizationHeader = request.getHeader(
        String tokenValue = authorizationHeader != null
        System.out.println("토큰값 = " + tokenValue);

        // 요청 본문에서 provider 추출
        String provider = body.get("provider");
        System.out.println("제공업체 = " + provider);

        //checkToken의 세번째 매개변수(허용된 역할 리스트) ->
        List<String> allowedRoles = Arrays.asList("USER

        // checkToken의 첫번째 매개변수(member 인스턴스 가져오
        TokenFactory tokenFactory = applicationContext.
        Map<String, Object> decodedToken = tokenFactory
        String oauth2ID =  String.valueOf(decodedToken.

        // 추상클래스를 통해 구현된 토큰팩토리를 가져옵니다.



        // Member member, String tokenValue, List allow
        // 토큰 유효성 검사

        Boolean isChecked = tokenFactory.checkToken(tok
        Boolean isExpired = tokenFactory.isExpired(Stri

        System.out.println("토큰이 유효합니까? " + isChecke
        System.out.println("토큰이 만료되었습니까? " + isExp

        if (!isChecked) { //토큰이 유효하지 않으면 삭제
            tokenFactory.deleteToken(oauth2ID);
            System.out.println("토큰이 유효하지 않아 삭제되었
            return ResponseEntity.ok()
                    .body(Map.of(
                        "message", "로그아웃 성공"
                    ));
        } else {
            // 토큰이 유효해도 만료와 상관없이 로그아웃 로직이므로
```

```
            tokenFactory.deleteToken(oauth2ID);
            System.out.println("로그아웃을 위해 토큰이 삭제되
            return ResponseEntity.ok()
                    .body(Map.of(
                        "message", "로그아웃 성공"
                    ));
        }
    }
```

2. 백엔드 디렉토리 재정리

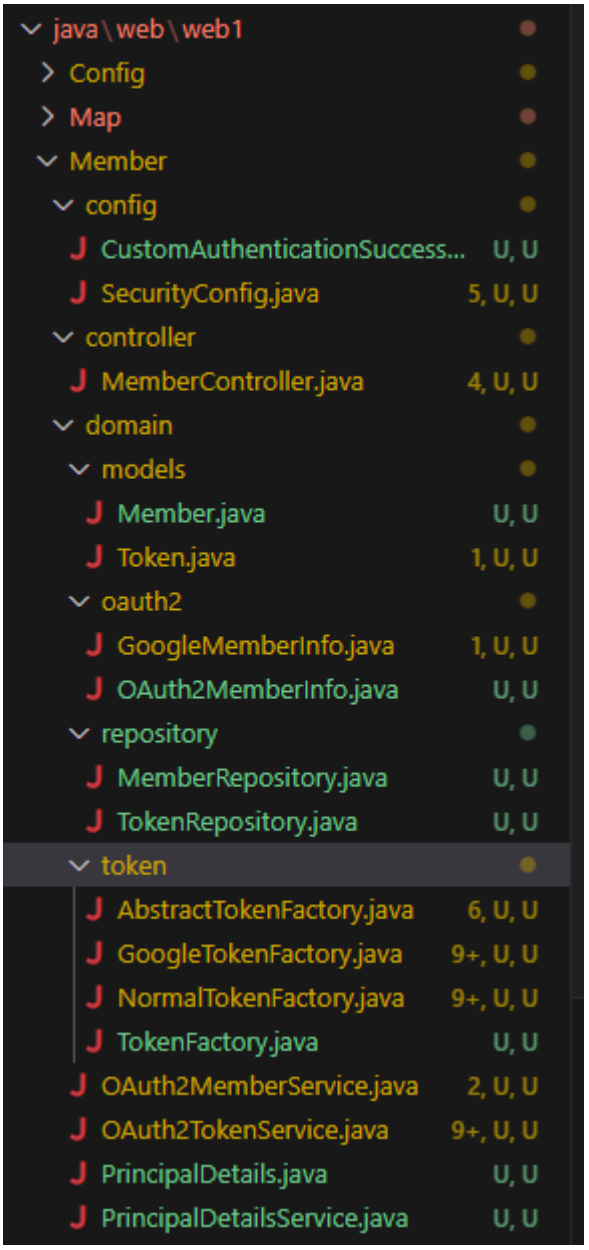
▼ 백엔드 디렉토리가 너무 어질러져 있어서 재정리와 리팩토링을 걸쳤다.

- 기존 디렉토리



- Oauth 디렉토리 안에
  - config
  - controller
  - domain
  - oauth2

        - token
    - 특히 domain 디렉토리의 .java들은 한눈에 어떤 역할을 수행하는지 잘 파악하기 힘들어서 수행역할별로 정리했다.



    - Config (프로젝트 전반의 설정 디렉토리)
    - Map (맵 관련 컴포넌트 디렉토리)
    - Member (유저 관련 컴포넌트 디렉토리)
        - config(Member컴포넌트의 데이터 흐름 관장 및 설정)
            - SecurityConfig

        - CustomAuthenticationSuccessHandler
    - controller(프론트로의 request와 response의 전체적인 흐름 관장)
        - MemberController (일반로그인 관련 로직을 어느정도 구현함)
    - domain
        - models (데이터베이스 디렉토리)
        - oauth2 (소셜로그인관련 기본 데이터의 명세 디렉토리)
        - token (토큰관련 기본 데이터의 명세와 로직구현 디렉토리)
        - repository (멤버와 토큰관련 JPA를 수행하기 위한 디렉토리)
        - 나머지 자바파일(소셜로그인 관련 로직구현)

3. SearchHistory 클래스 구현

▼ SearchHistory (유저 검색관련 로직구현함)

- 유저검색기록에 대해 CRUD를 수행함

```
package web.web1.Map.domain.searchhistory;

import java.time.LocalDateTime;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.Collections;

import org.springframework.beans.factory.annotation.Autowi
import org.springframework.jdbc.core.JdbcTemplate;

import web.web1.Member.domain.models.Member;
import web.web1.Member.domain.token.TokenFactory;


public class SearchHistory {

    private TokenFactory tokenFactory;
```

```
    private final JdbcTemplate jdbcTemplate;

    @Autowired
    public SearchHistory(JdbcTemplate jdbcTemplate, TokenF
        this.tokenFactory = tokenFactory;
        this.jdbcTemplate = jdbcTemplate;
    }


    public List<Map<String, Object>> getHistory(String tok

        // 이 메소드는 user와 admin인 경우 실행 가능함
        List<String> allowedRoles = Arrays.asList("USER",

        Map<String, Object> decodedToken = tokenFactory.de
        String oauth2Id = (String) decodedToken.get("oauth

        Boolean isChecked = tokenFactory.checkToken(tokenV
        Boolean isExpired = tokenFactory.isExpired(String.

        if (isChecked && !isExpired){
            System.out.println("당신의 토큰은 이 작업을 실행하기

            // SQL 시작
            String sql1 = "SELECT email FROM member WHERE
            String email = jdbcTemplate.queryForObject(sql

            if (email != null) {
                String sql2 = "SELECT * FROM memberhistory
                List<Map<String, Object>> historyList = jd
                return historyList;
            } else {
                System.out.println("당신의 이메일이 존재하지 않
                return Collections.emptyList();
            }
        }
        else {
            System.out.println("당신의 토큰은 이 작업을 실행하기
```

```
            return null;
        }

    }

    public void createHistory(String tokenValue, String se

        // 이 메소드는 user와 admin인 경우 실행 가능함
        List<String> allowedRoles = Arrays.asList("USER",

        Map<String, Object> decodedToken = tokenFactory.de
        String oauth2Id = (String) decodedToken.get("oauth

        Boolean isChecked = tokenFactory.checkToken(tokenV
        Boolean isExpired = tokenFactory.isExpired(String.

        if (isChecked && !isExpired){
            System.out.println("당신의 토큰은 이 작업을 실행하기

            // oauth2Id를 이용해 사용자 이메일 조회
            String sql1 = "SELECT email FROM member WHERE
            String email = jdbcTemplate.queryForObject(sql
            if (email != null) {
                // 현재 시간을 검색 기록의 시간으로 사용
                LocalDateTime searchDateTime = LocalDateTi
                String sql2 = "INSERT INTO memberhistory (
                jdbcTemplate.update(sql2, email, searchQue
                System.out.println("검색 기록이 추가되었습니다."
            } else {
                System.out.println("사용자 정보를 찾을 수 없습니
            }
        } else {
            System.out.println("당신의 토큰은 이 작업을 실행하기
        }
    }


    public void deleteHistory(String tokenValue) {
```

```
        List<String> allowedRoles = Arrays.asList("USER",

        Map<String, Object> decodedToken = tokenFactory.de
        String oauth2Id = (String) decodedToken.get("oauth

        Boolean isChecked = tokenFactory.checkToken(tokenV
        Boolean isExpired = tokenFactory.isExpired(String.

        if (isChecked && !isExpired){
            System.out.println("당신의 토큰은 이 작업을 실행하기

            String sql1 = "SELECT email FROM member WHERE
            String email = jdbcTemplate.queryForObject(sql
            if (email != null) {
                String sql2 = "DELETE FROM memberhistory W
                int rowsAffected = jdbcTemplate.update(sql
                System.out.println(rowsAffected + " rows d
            } else {
                System.out.println("사용자 정보를 찾을 수 없습니
            }
        }
    }
}
```

4. 프론트엔드 UI 개선

▼ 프론트엔드 UI를 개선함.

기존 거의 빈화면이었던 메인페이지를 개선했다.

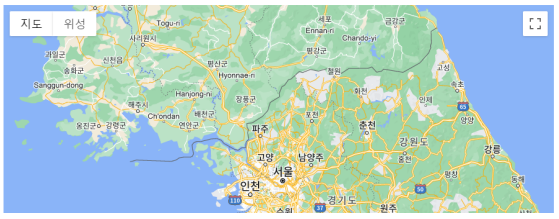
Didim365

스프링 기반 도로명주소 찾기 서비스

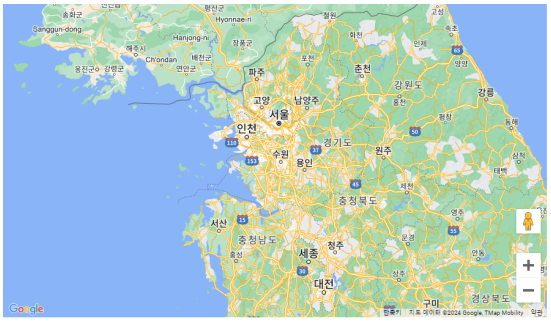
login

신촌

주소 찾기



도로명 주소	지번 주소	우편번호	버튼
울산광역시 울주군 청량읍 신촌1길 2	울산광역시 울주군 청량읍 용암리 689-2 신촌마을회관	44989	Select
강원특별자치도 횡성군 공근면 신촌길 51	강원특별자치도 횡성군 공근면 신촌리 607-4 신촌리노인회	25209	Select
전북특별자치도 고창군 고창읍 신촌2길 58	전북특별자치도 고창군 고창읍 율계리 372-11 신촌경로복지회관	56442	Select
전북특별자치도 고창군 공음면 장곡신촌길 37	전북특별자치도 고창군 공음면 장곡리 199 장곡리신촌마을회관	56460	Select
전북특별자치도 고창군 무장면 신촌길 47	전북특별자치도 고창군 무장면 신촌리 303-1 신촌노인정	56458	Select
전북특별자치도 고창군 신림면 자포신촌길 4	전북특별자치도 고창군 신림면 자포리 600-4 신촌마을회관	56424	Select
전북특별자치도 군산시 나포면 주곡신촌길 15	전북특별자치도 군산시 나포면 주곡리 129 신촌마을경노당	54049	Select
전북특별자치도 군산시 대야면 복교신촌길 34-4	전북특별자치도 군산시 대야면 복교리 944-52 신촌경로당	54062	Select
전북특별자치도 남원시 신촌길 5 (신촌동)	전북특별자치도 남원시 신촌동 224-4 신촌경로당	55794	Select



강원특별자치도 횡성군 공근면 신촌길 51	강원특별자치도 횡성군 공근면 신촌리 607-4 신촌리노인회	25209	Select
전북특별자치도 고창군 고창읍 신촌2길 58	전북특별자치도 고창군 고창읍 율계리 372-11 신촌경로복지회관	56442	Select
전북특별자치도 고창군 공음면 장곡신촌길 37	전북특별자치도 고창군 공음면 장곡리 199 장곡리신촌마을회관	56460	Select
전북특별자치도 고창군 무장면 신촌길 47	전북특별자치도 고창군 무장면 신촌리 303-1 신촌노인정	56458	Select
전북특별자치도 고창군 신림면 자포신촌길 4	전북특별자치도 고창군 신림면 자포리 600-4 신촌마을회관	56424	Select
전북특별자치도 군산시 나포면 주곡신촌길 15	전북특별자치도 군산시 나포면 주곡리 129 신촌마을경노당	54049	Select
전북특별자치도 군산시 대야면 복교신촌길 34-4	전북특별자치도 군산시 대야면 복교리 944-52 신촌경로당	54062	Select
전북특별자치도 남원시 신촌길 5 (신촌동)	전북특별자치도 남원시 신촌동 224-4 신촌경로당	55794	Select
전북특별자치도 남원시 인월면 인월신촌길 11	전북특별자치도 남원시 인월면 서무리 624-9 신촌새마을회관	55715	Select
전북특별자치도 무주군 안성면 신촌길 21	전북특별자치도 무주군 안성면 장기리 1308-100 신촌새마을회관,경로당	55540	Select
전북특별자치도 순창군 순창읍 신촌길 102	전북특별자치도 순창군 순창읍 백산리 72-2 신촌마을회관	56048	Select
전북특별자치도 익산시 춘포면 신촌1길 24	전북특별자치도 익산시 춘포면 춘포리 120 신촌경로당	54581	Select

1 2 3 4 5 6 7 8

앞으로 해야되는것

로그인 마이페이지 구현

유저 추천시스템 구현