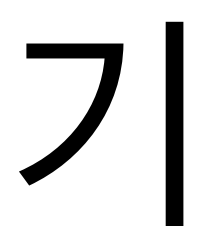
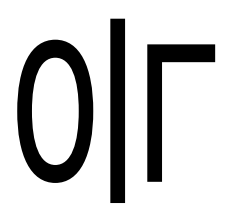
Social Login

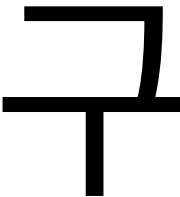
Response Header

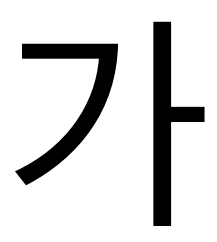
```
export const __userOauthKakao = createAsyncThunk(
  "oauth/USER_OAUTH_KAKAO",
  async (payload, thunkAPI) => {
    try{
      const data = await api.post(`auth/kakao`, {payload})
      .then((res)=>{
        const accessToken = res.headers.authorization;
        const refreshToken = res.headers.refreshtoken;
        const nickname = res.data.nickname;

        //유저 토큰 + 닉네임이 있다면 가져온 후 세팅
        if(accessToken && refreshToken && nickname){
          localStorage.setItem("token", accessToken);
          localStorage.setItem("refreshToken", refreshToken);
          localStorage.setItem("nickname", nickname);
        }else{
          alert('인증 오류! 다시 시도해주세요!')
           return window.location.assign("/");
        }
        //유저 토큰 + 닉네임 가져오기
        const accessTokenGet = localStorage.getItem("token");
        const refreshTokenGet = localStorage.getItem("refreshToken");
        const nicknameGet = localStorage.getItem("nickname");
        if(accessTokenGet && refreshTokenGet && nicknameGet){
          alert('소셜로그인 인증 완료!')
          window.location.assign("/main");
        }else{
          alert('연결 오류! 다시 시도해주세요!')
          return window.location.assign("/");
        }
        return res
      })
      return thunkAPI.fulfillWithValue(data)
    }catch(error){
      window.location.assign("/");
      return thunkAPI.rejectWithValue(error)
    }
  }
);
```

당

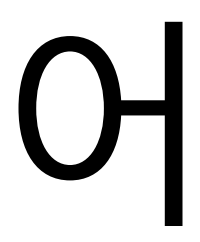


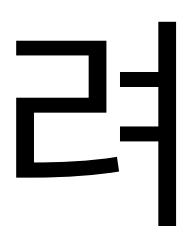

o|r



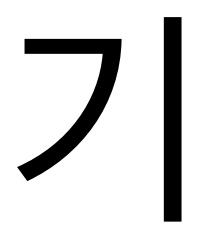


장

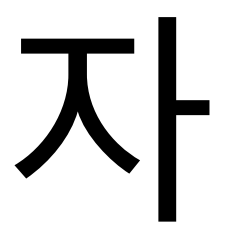


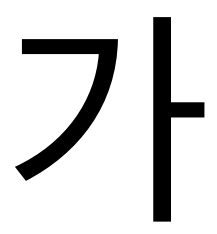


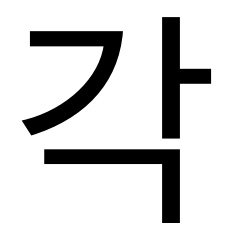
원

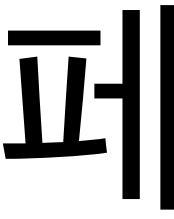


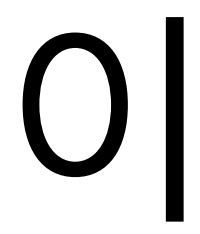

o|r

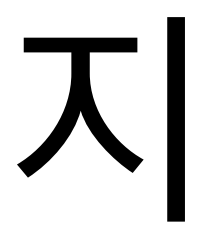




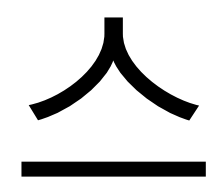


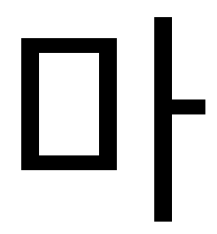


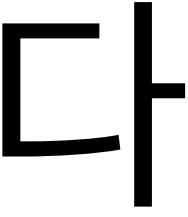




서

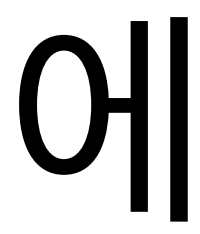


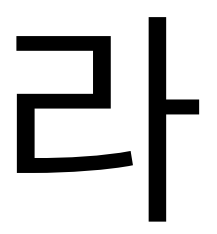


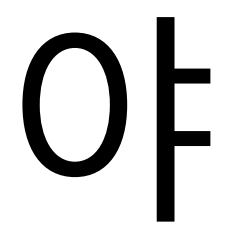


정

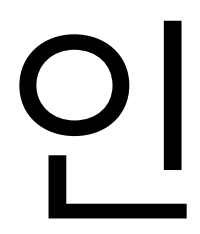
식

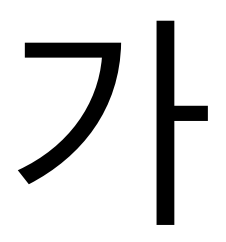


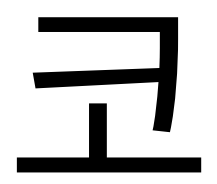


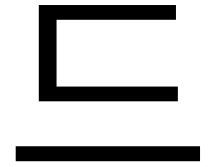


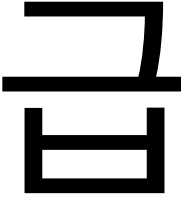

함



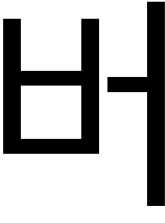

가

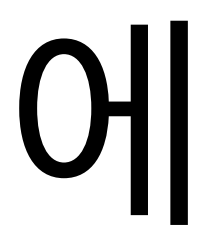


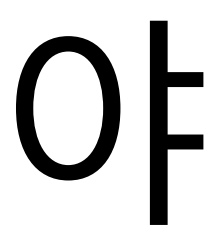


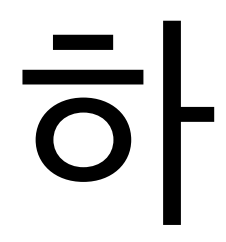


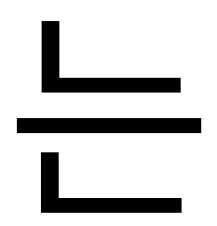
서



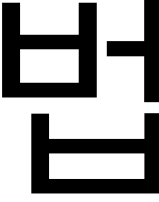


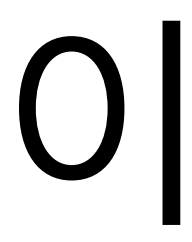


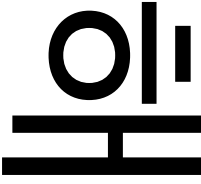


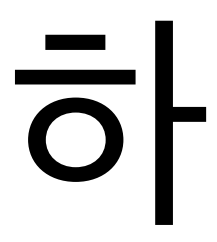


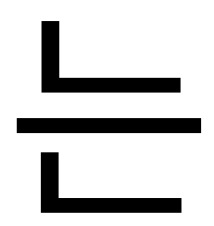
방

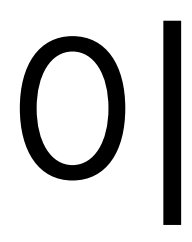

법

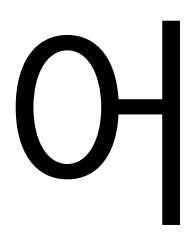


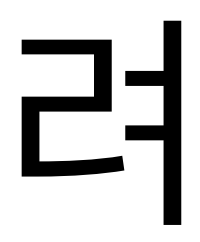


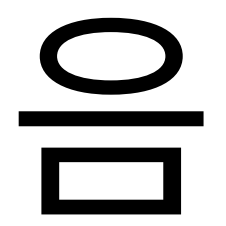


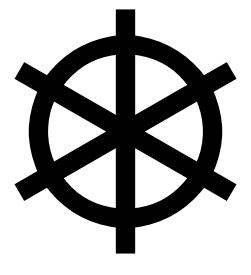


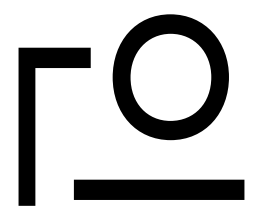






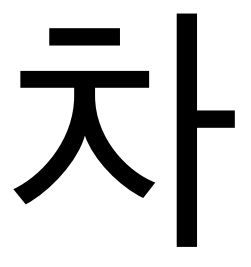


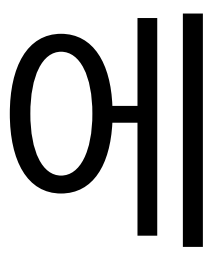


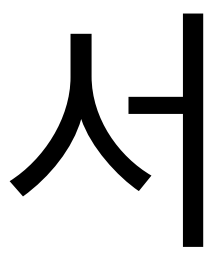


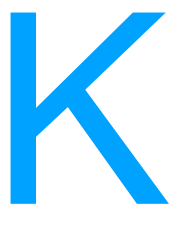

klo


절

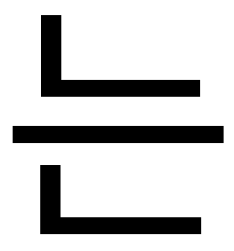

차

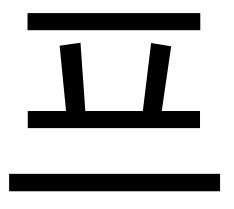


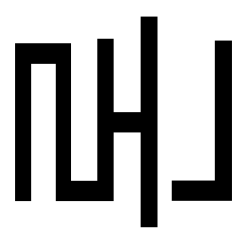


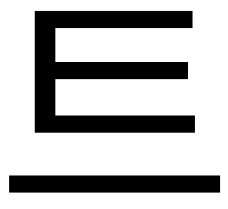



y

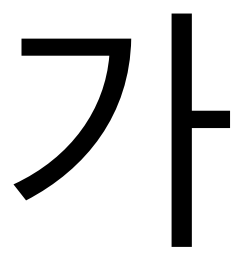


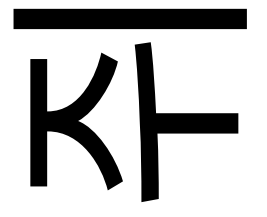


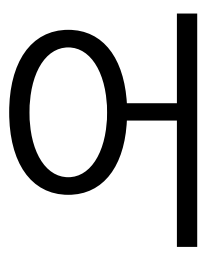


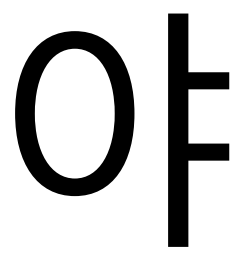


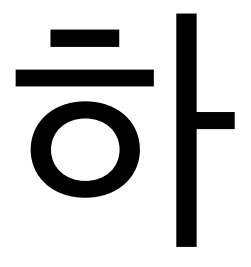

엔

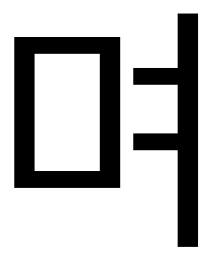


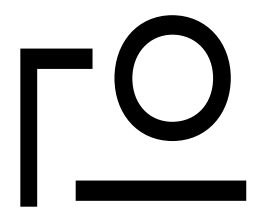


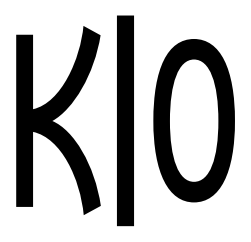


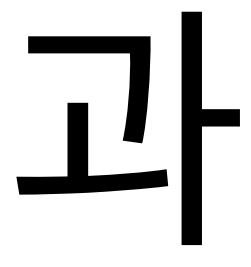




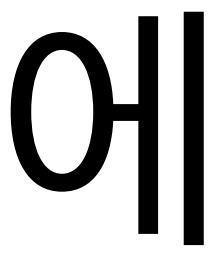


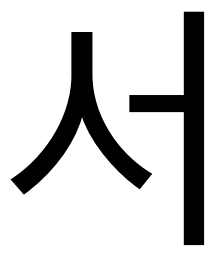


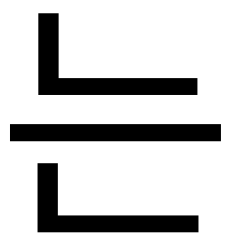

KIO

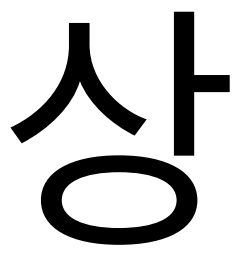

과

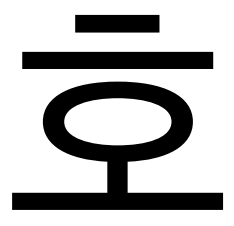
정



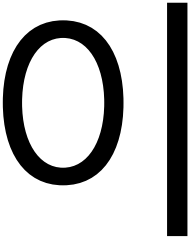




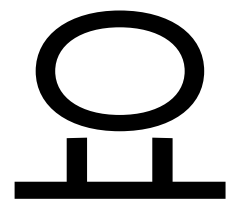

상

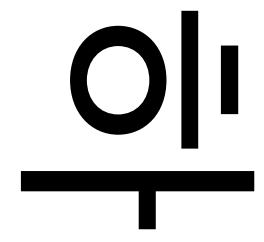


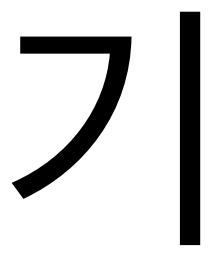

OHM

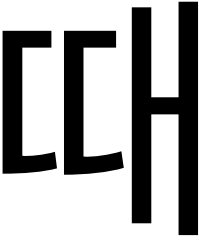


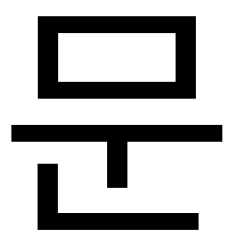
중

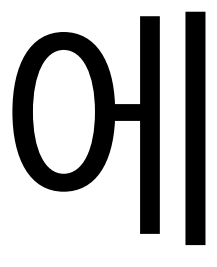


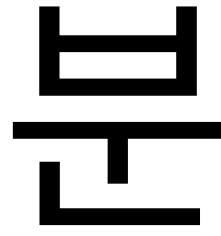




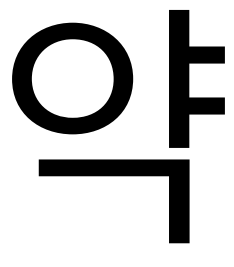

CCH

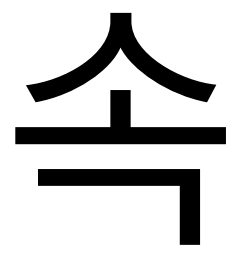


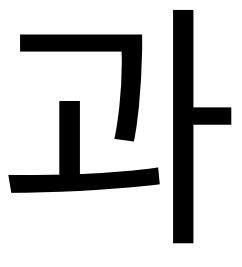


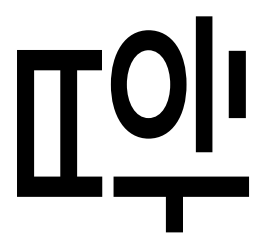


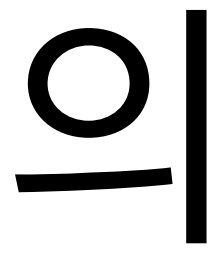
한



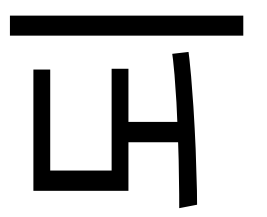


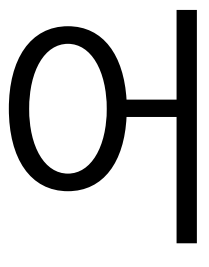


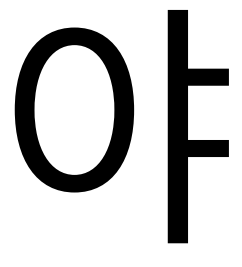




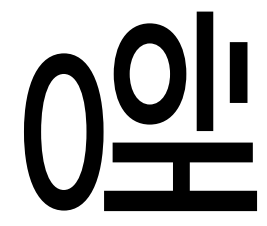
フト

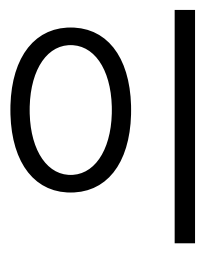


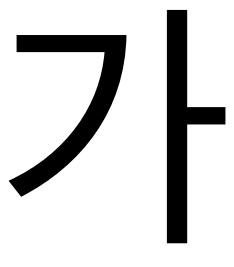


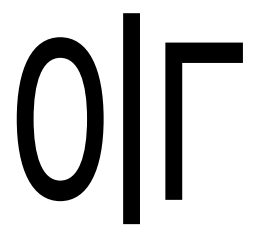


진






フト



해당 기능구현 중 가장 어려웠던 것

- 기능자체가 각 페이지 및 서비스마다 정해진 양식에 따라야함
- 인가 코드 발급 및 서버에 전달을 해야하는 방법을 이해하는것이 어려웠음

Social Login

Response Header

```
export const __userOauthKakao = createAsyncThunk(
  "oauth/USER_OAUTH_KAKAO",
  async (payload, thunkAPI) => {
    try{
      const data = await api.post(`auth/kakao`, {payload})
      .then((res)=>{
        const accessToken = res.headers.authorization;
        const refreshToken = res.headers.refreshtoken;
        const nickname = res.data.nickname;

        //유저 토큰 + 닉네임이 있다면 가져온 후 세팅
        if(accessToken && refreshToken && nickname){
          localStorage.setItem("token", accessToken);
          localStorage.setItem("refreshToken", refreshToken);
          localStorage.setItem("nickname", nickname);
        }else{
          alert('인증 오류! 다시 시도해주세요!')
           return window.location.assign("/");
        }
        //유저 토큰 + 닉네임 가져오기
        const accessTokenGet = localStorage.getItem("token");
        const refreshTokenGet = localStorage.getItem("refreshToken");
        const nicknameGet = localStorage.getItem("nickname");
        if(accessTokenGet && refreshTokenGet && nicknameGet){
          alert('소셜로그인 인증 완료!')
          window.location.assign("/main");
        }else{
          alert('연결 오류! 다시 시도해주세요!')
          return window.location.assign("/");
        }
        return res
      })
      return thunkAPI.fulfillWithValue(data)
    }catch(error){
      window.location.assign("/");
      return thunkAPI.rejectWithValue(error)
    }
  }
);
```

해당 기능구현 중 가장 어려웠던 것

- 기능자체가 각 페이지 및 서비스마다 정해진 양식에 따라야함
- 인가 코드 발급 및 서버에 전달을 해야하는 방법을 이해하는것이 어려웠음

❄ 인증 절차에서 Key는 프론트엔드가 쥐어야 하며,
인증 과정에서는 상호통신이 중요하기 때문에
충분한 약속과 합의가 되어야 진행이 가능

1조의 발표는 이상입니다.

감사합니다.