Kakao Developer Buddies

블로그 프로토타입

목차 A table of contents

1 WBS

2 FE 구현

3 FE 페이지 디자인

4 데이터베이스 디자인

5 BE 구현

6 Trouble Shooting

7 팀원 소개

1

WBS

1 WBS

업무 목록

요약	상태	범주	담당자	기한	우...
기존 제품 분석	완료됨	기획	Hogi Chai	2023년 9월 25일	
서론 작성	완료됨	기획	장현화	2023년 9월 29일	
UI / UX 디자인	완료됨	기획	윤건우	2023년 10월 2일	
스토리보드 작성	완료됨	기획	강석훈	2023년 10월 3일	
요구사항 정의	완료됨	기획	김성민	2023년 10월 3일	
컴포넌트 구성	완료됨	기획	윤건우	2023년 10월 1일	
Use-Case Diagram 작성	완료됨	기획	김성민	2023년 10월 2일	
Sequence Diagram 작성	완료됨	기획	윤건우	2023년 10월 2일	
State Diagram 작성	완료됨	기획	Hogi Chai	2023년 10월 2일	
프론트엔드 분석 & 설계	진행중	개발	강석훈	2023년 10월 27일	
백엔드 분석 & 설계	진행중	개발	윤건우	2023년 10월 27일	
프론트 & 백엔드 통합	해야 할 일	개발	윤건우	2023년 10월 31일	
아키텍처 설계	진행중	개발	김성민	2023년 10월 29일	
프로토타입 발표 준비	해야 할 일	발표	Hogi Chai	2023년 10월 31일	
프로토타입 발표	해야 할 일	발표	김성민	2023년 10월 31일	

요약	상태	범주	담당자	기한	우...
프론트엔드 1차 스프린트	해야 할 일	개발	강석훈	2023년 11월 7일	
백엔드 1차 스프린트	해야 할 일	개발	윤건우	2023년 11월 7일	
프론트엔드 2차 스프린트	해야 할 일	개발	강석훈	2023년 11월 14일	
백엔드 2차 스프린트	해야 할 일	개발	윤건우	2023년 11월 14일	
1차 QA	해야 할 일	QA	윤건우	2023년 11월 15일	
1차 수정 & 개발	해야 할 일	QA	윤건우	2023년 11월 20일	
2차 QA	해야 할 일	QA	윤건우	2023년 11월 22일	
2차 수정 & 개발	해야 할 일	QA	윤건우	2023년 11월 27일	
최종 배포	해야 할 일	QA	윤건우	2023년 11월 30일	
회고 작성	해야 할 일	마무리	강석훈	2023년 12월 2일	
문서화	해야 할 일	마무리	장현화	2023년 12월 6일	
최종 발표 준비	해야 할 일	발표	Hogi Chai	2023년 12월 13일	
최종 발표	해야 할 일	발표	김성민	2023년 12월 14일	

1 WBS

타임라인

항목
9월
10월
11월
KADB-8 기존 제품 분석
KADB-9 서론 작성
KADB-10 UI / UX 디자인
KADB-11 스토리보드 작성
KADB-12 요구사항 정의
KADB-13 컴포넌트 구성
KADB-14 Use-Case Diagram 작성
KADB-15 Sequence Diagram 작성
KADB-16 State Diagram 작성
KADB-18 프론트엔드 분석 & 설계
KADB-19 백엔드 분석 & 설계
KADB-21 아키텍처 설계
KADB-20 프론트 & 백엔드 통합
KADB-22 프로토타입 발표 준비
KADB-23 프로토타입 발표

1 WBS

타임라인

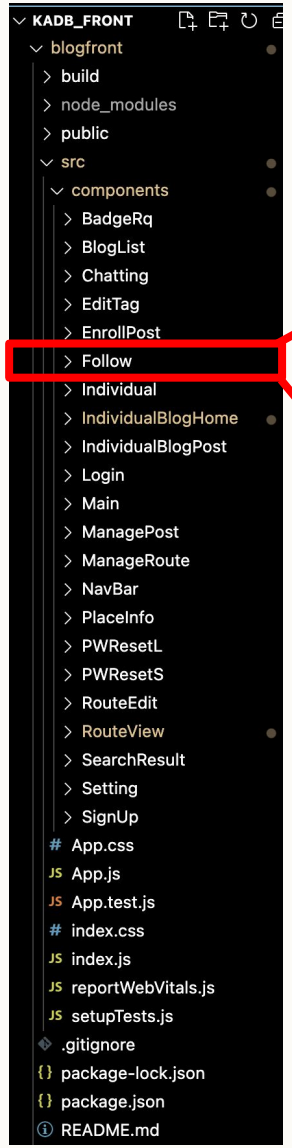
항목
9월
10월
11월
12월
KADB-24 프론트엔드 1차 스프린트
KADB-25 백엔드 1차 스프린트
KADB-26 프론트엔드 2차 스프린트
KADB-27 백엔드 2차 스프린트
KADB-28 1차 QA
KADB-29 1차 수정 & 개발
KADB-30 2차 QA
KADB-31 2차 수정 & 개발
KADB-32 최종 배포
KADB-33 회고 작성
KADB-34 문서화
KADB-35 최종 발표 준비
KADB-36 최종 발표

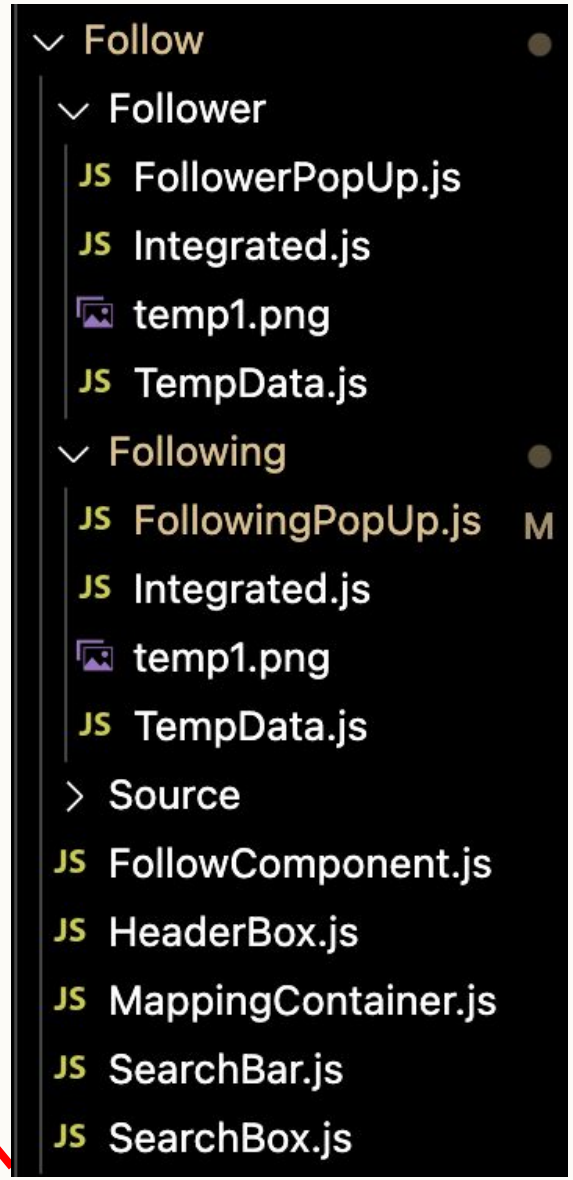
2

FE 구현

2 FE 구현

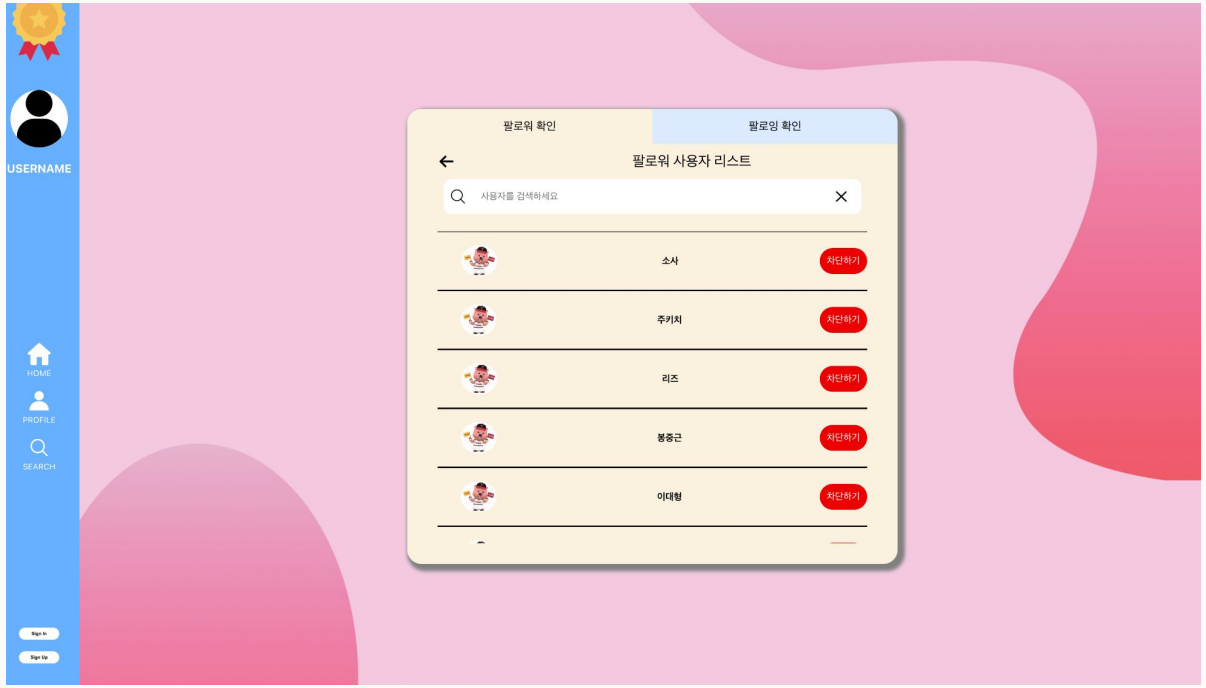
기본 구조

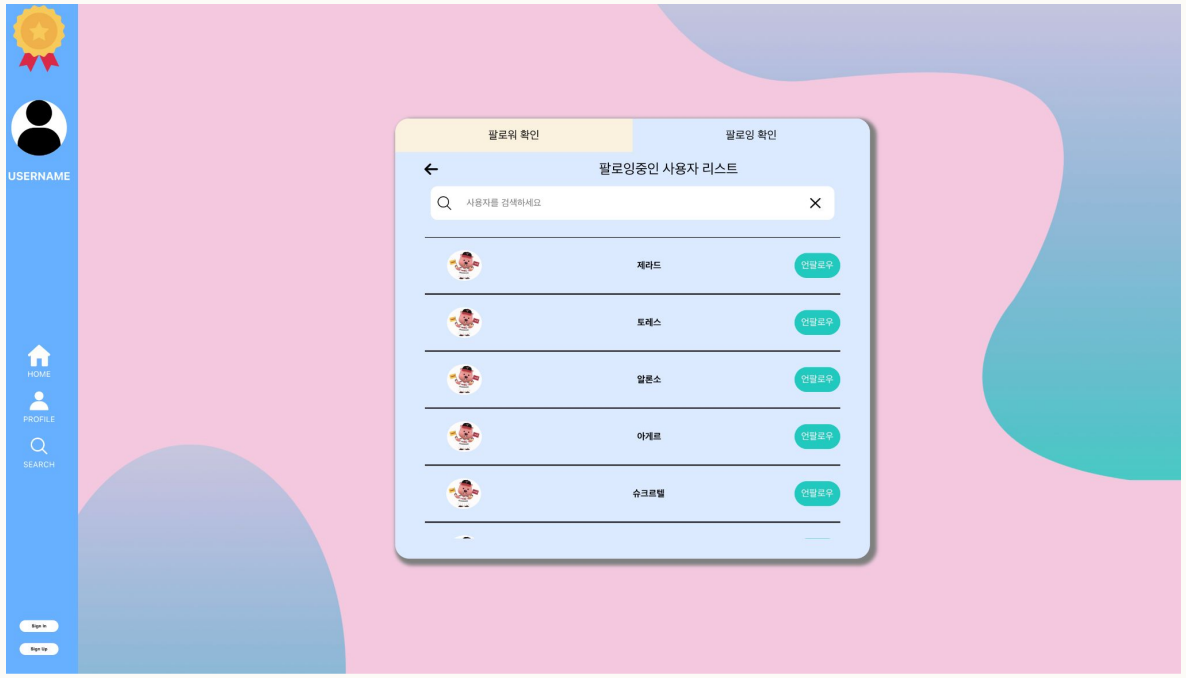

KADB_FRONT
blogfront
build
node_modules
public
src
components
BadgeRq
BlogList
Chatting
EditTag
EnrollPost
Follow
Individual
IndividualBlogHome
IndividualBlogPost
Login
Main
ManagePost
ManageRoute
NavBar
PlaceInfo
PWResetL
PWResetS
RouteEdit
RouteView
SearchResult
Setting
SignUp
App.css
App.js
App.test.js
index.css
index.js
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md


Follow
Follower
FollowerPopUp.js
Integrated.js
temp1.png
TempData.js
Following
FollowingPopUp.js
M
Integrated.js
temp1.png
TempData.js
Source
FollowComponent.js
HeaderBox.js
MappingContainer.js
SearchBar.js
SearchBox.js

2 FE 구현

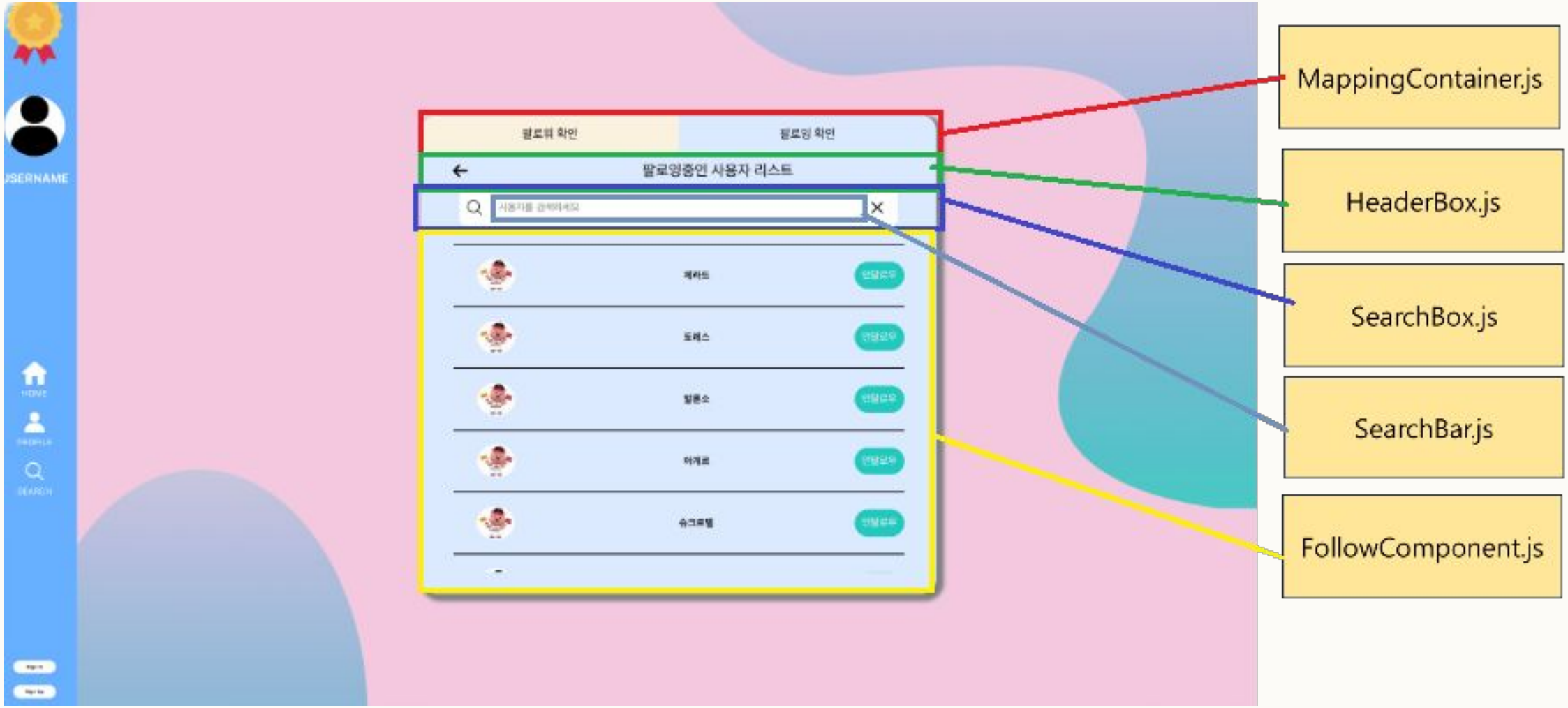
페이지 예시


USERNAME
HOME
PROFILE
SEARCH
팔로워 확인
팔로잉 확인
팔로워 사용자 리스트
사용자를 검색하세요
소사
주키치
리즈
봉중근
이대형
차단하기


USERNAME
HOME
PROFILE
SEARCH
팔로워 확인
팔로잉 확인
팔로잉중인 사용자 리스트
사용자를 검색하세요
제라드
토레스
알론소
아게르
슈크르텔
언팔로우

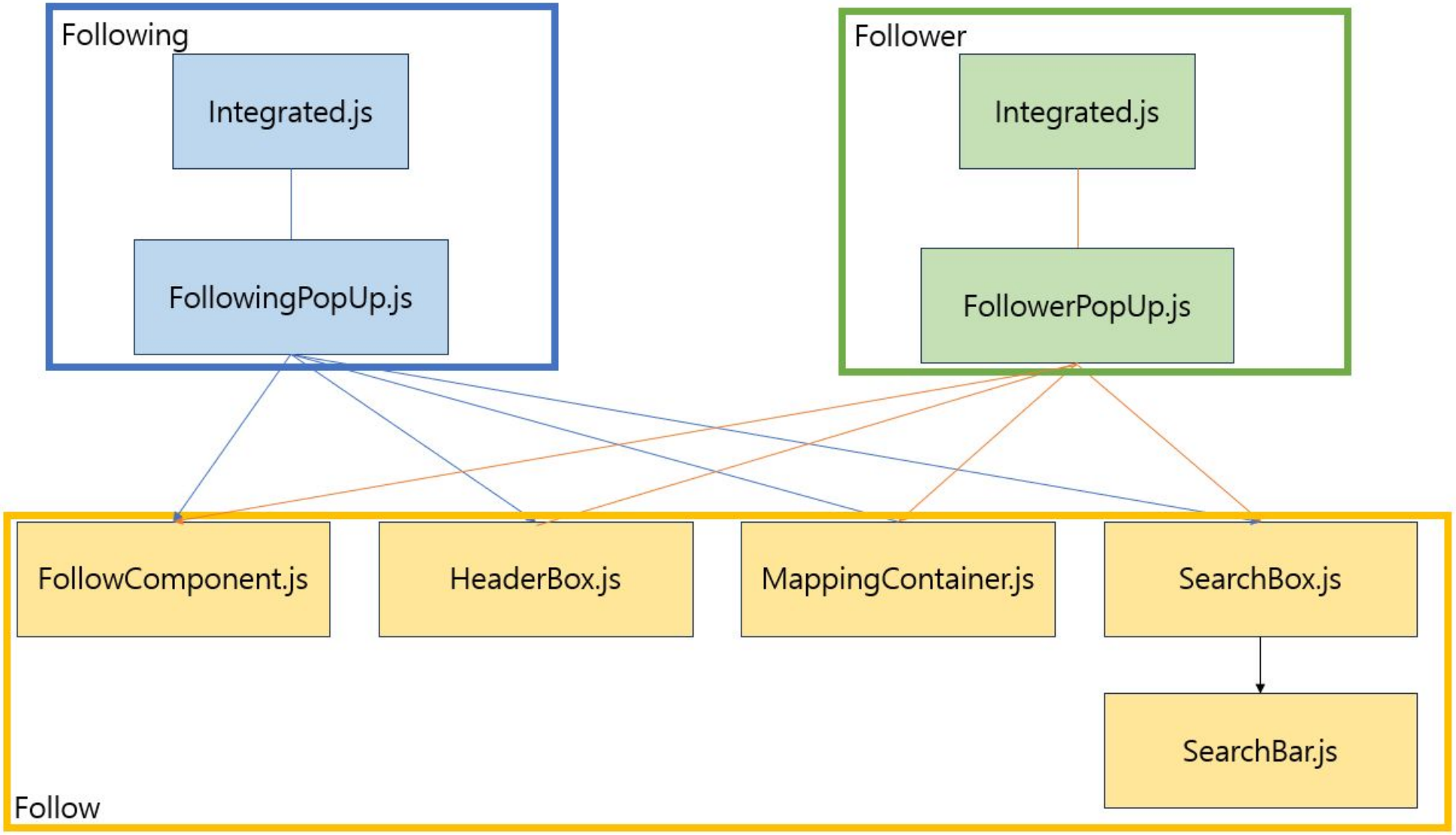
2 FE 구현

Component 이용 - 코드 재사용



2 FE 구현

Component 이용 - 코드 재사용



©Saebyeol Yu. Saebyeol's PowerPoint

2 FE 구현

Component 이용 - 코드 재사용

```
import HeaderBox from '../HeaderBox'
import MappingBox from '../MappingContainer';
import SearchBox from '../SearchBox';
import FollowComponent from '../FollowComponent';
```

```
return (
    <>
        <Container>
            <FollowerBox>
                <MappingBox />
                <HeaderContainer>
                    <HeaderBox text="팔로잉중인 사용자 리스트"/>
                    <SearchBox onChange={handleSearchChange} onClear={handleClearSearch} />
                </HeaderContainer>
                <FollowerComponentContainer>
                    <FollowComponent data={userData} colour="#23C9BF"
                        onButtonClick={handleBlockUser} text="언팔로우" />
                </FollowerComponentContainer>
            </FollowerBox>
        </Container>
    </>
);
```

FollowingPopUp.js

```
function FollowComponent(props) {

    return (
        props.data.map(function (a) {
            return (
                <FollowerContainer>
                    <ImgBox>
                        <ImgSrc src={a.img} alt={a.name} />
                    </ImgBox>
                    <NameBox>
                        <h5>{a.name}</h5>
                    </NameBox>
                    <Btn
                        bgcolour={props.colour}
                        onClick={() => {
                            const updatedData = props.data.filter(item => item.name !== a.name);
                            props.onButtonClick(updatedData);
                        }}>{props.text}</Btn>
                </FollowerContainer>
            );
        })
    );
}
```

FollowComponent.js

2 FE 구현

Component 이용 - 코드 재사용

```
import styled from 'styled-components';

let FollowerContainer = styled.div`

    border-top: 1px solid black;
    border-bottom: 1px solid black;
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
    width: 35vw;
    height: 80px;

`
```

Styled 이용 - CSS 이름 중복 충돌 방지

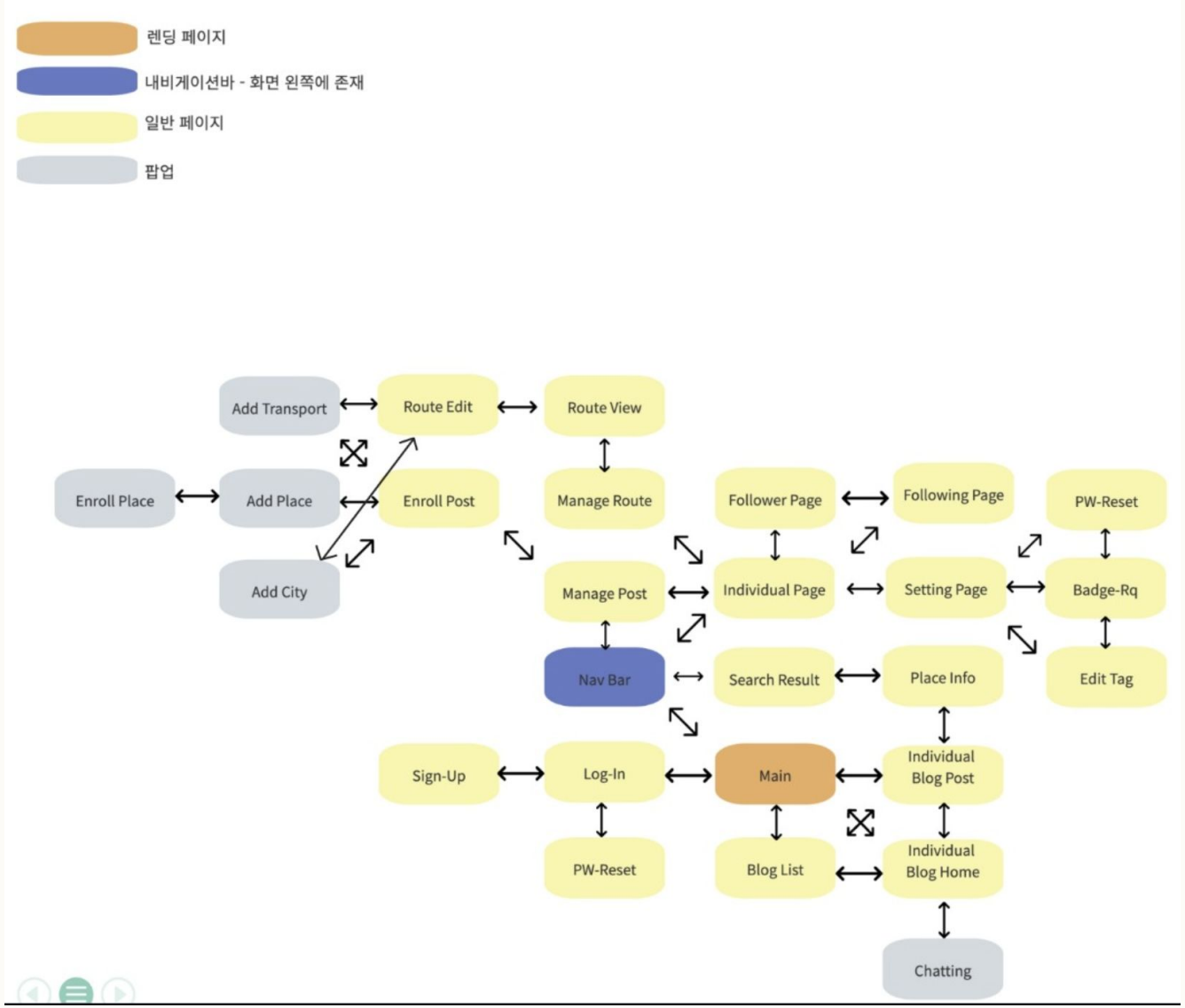
```
<React.StrictMode>
  <BrowserRouter>
    {/* NavBar 필요 있는 것들 */}
    <Routes>
      <Route path="/" element={<App />}>
        <Route path="main" element={<Main/>}/>
        <Route path="bloglist" element={<BlogList/>}/>
        <Route path="individual" element={<Individual/>}/>
        <Route path="placeinfo" element={<PlaceInfo/>}/>
        <Route path="bloghome" element={<IndividualBlogHome/>}/>
        <Route path="blogpost" element={<IndividualBlogPost/>}/>
        <Route path="setting" element={<Setting />} />
        <Route path="follower" element={<Follower />} />
        <Route path="following" element={<Following />} />
        <Route path="setting/pw" element={<PWResetS />} />
        <Route path="setting/bg" element={<BadgeRq />} />
        <Route path="setting/tag" element={<EditTag />} />
        <Route path="enrollpost" element={<EnrollPost/>}/>
        <Route path="managepost" element={<ManagePost/>}/>
        <Route path="routeview" element={<RouteView />} />
        <Route path="manageroute" element={<ManageRoute />}/>

      </Route>

      {/* NavBar 필요 없는 것들 */}
      <Route path="/login" element={<Login />} />
      <Route path="/signup" element={<SignUp />} />
      <Route path="/pwresetl" element={<PWResetL />} />
    </Routes>
  </BrowserRouter>
</React.StrictMode>
```

페이지간 전환 - routing이용

FE 페이지 구성

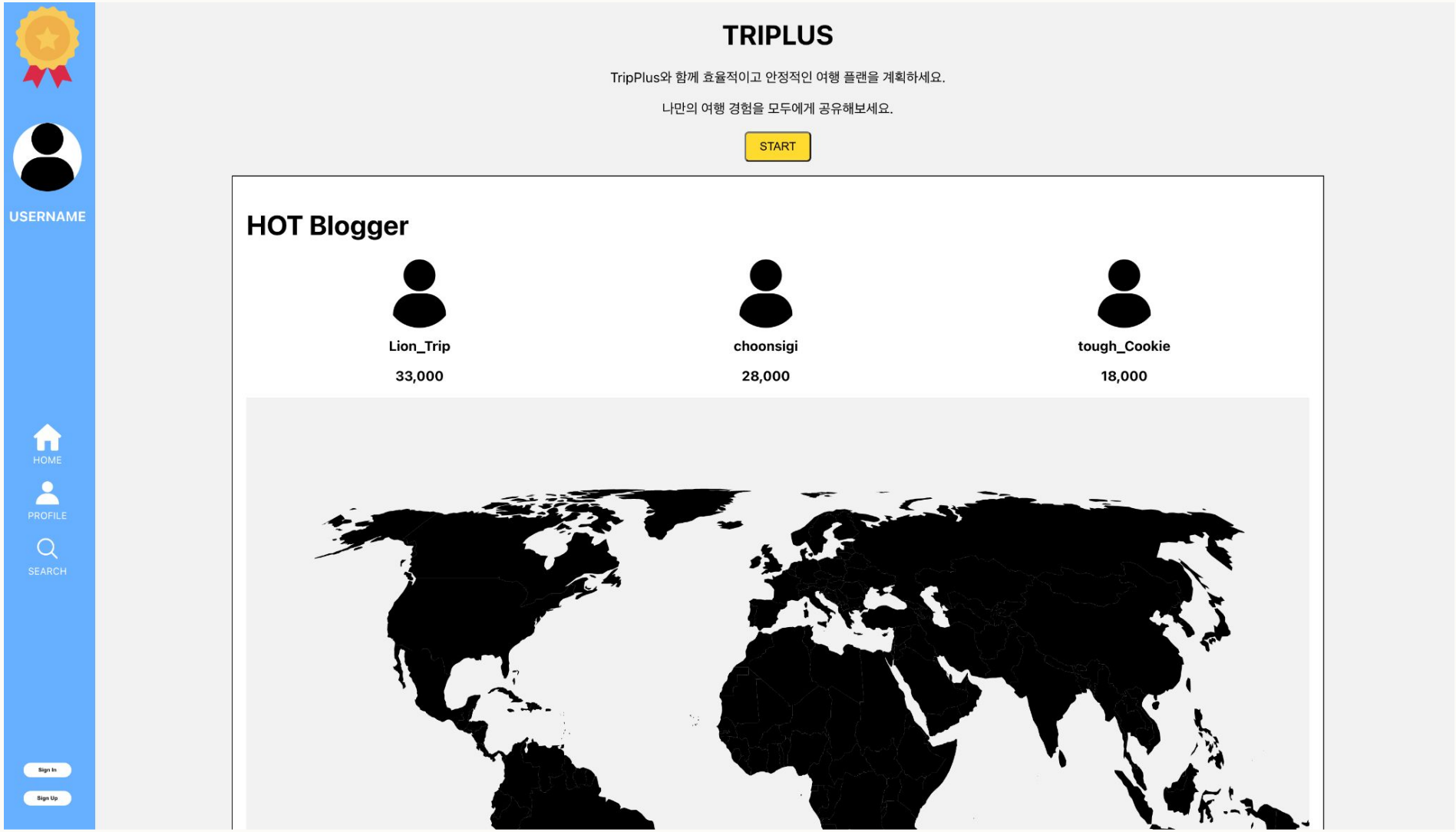
3

3 페이지 구성도


렌딩 페이지
내비게이션바 - 화면 왼쪽에 존재
일반 페이지
팝업
Add Transport
Route Edit
Route View
Enroll Place
Add Place
Enroll Post
Manage Route
Follower Page
Following Page
PW-Reset
Add City
Manage Post
Individual Page
Setting Page
Badge-Rq
Nav Bar
Search Result
Place Info
Edit Tag
Sign-Up
Log-In
Main
Individual Blog Post
PW-Reset
Blog List
Individual Blog Home
Chatting

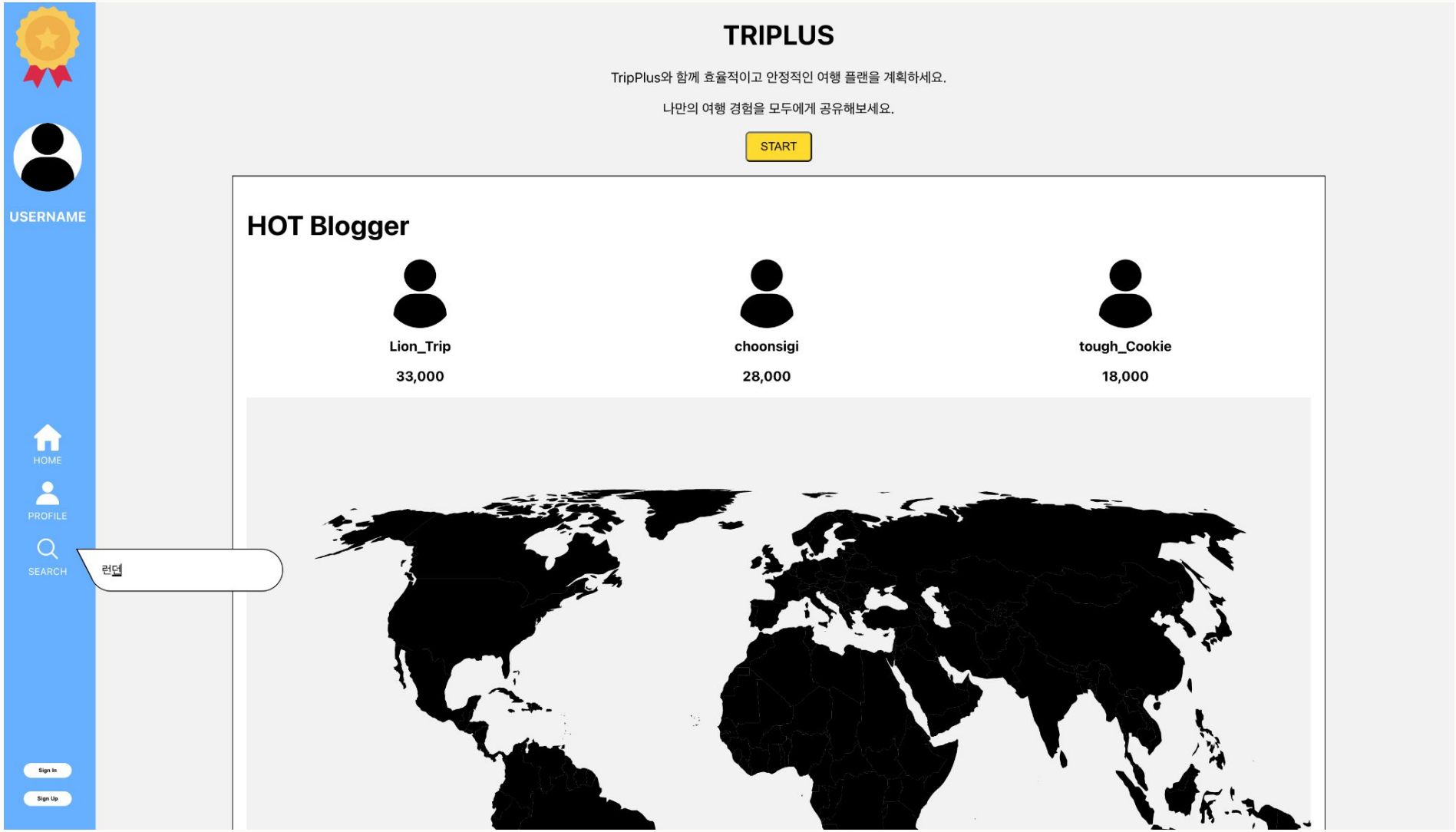
3 페이지 구성도

메인 페이지(랜딩 페이지)



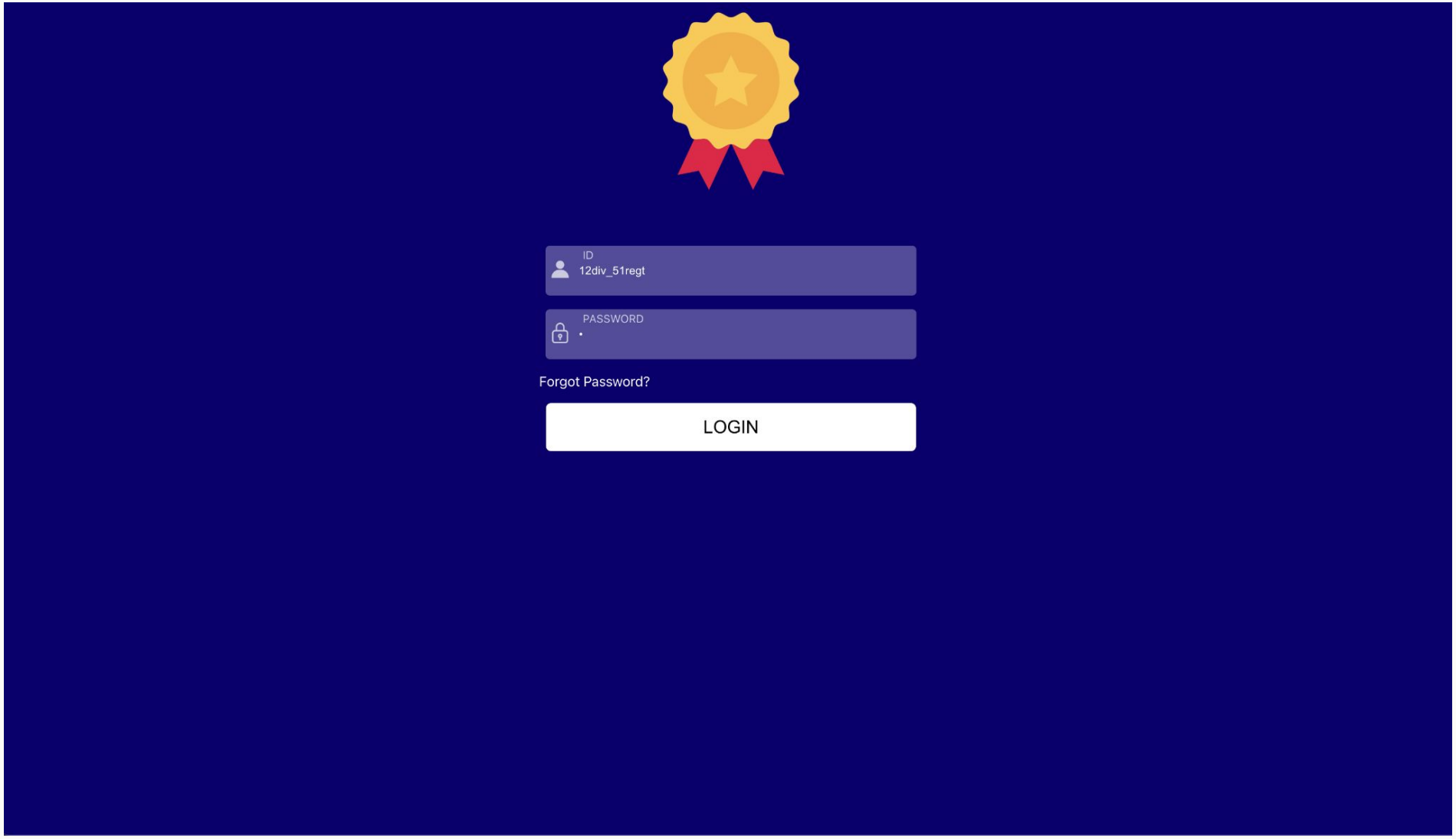
3 페이지 구성도

메인 페이지(랜딩 페이지) - 검색 기능



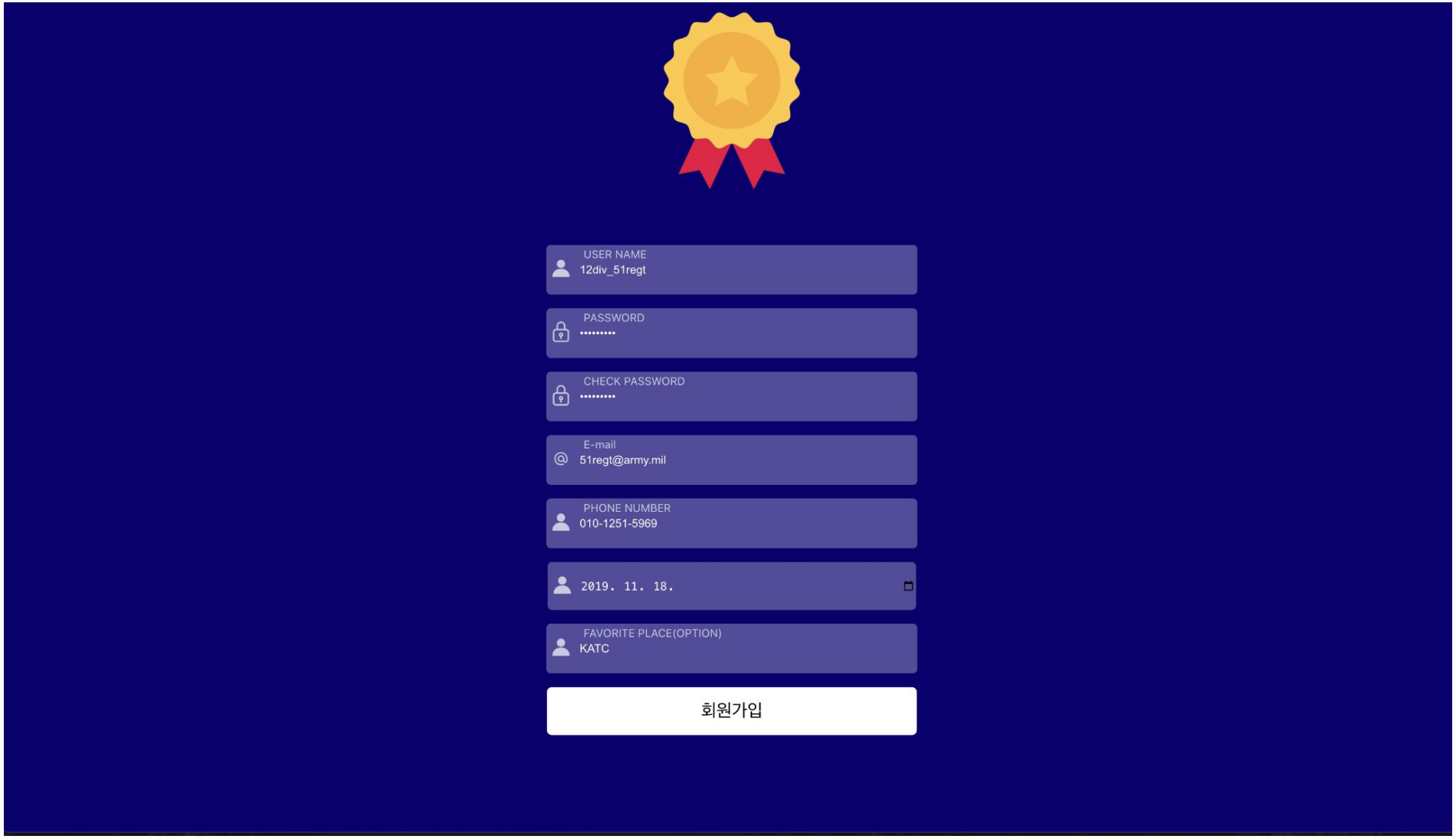
3 페이지 구성도

로그인 페이지



3 페이지 구성도

회원가입

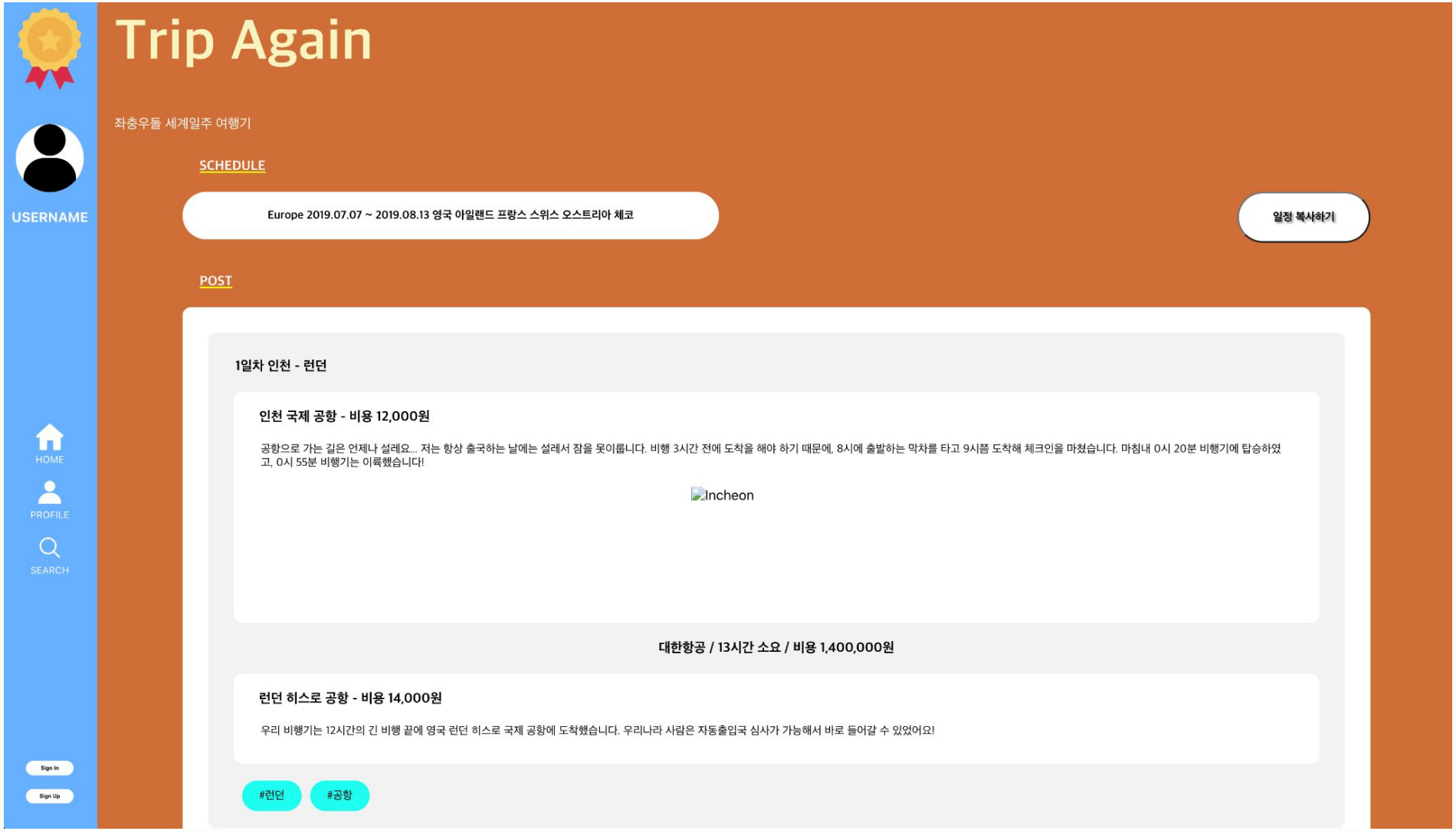


3 페이지 구성도

개인 블로그 메인 페이지

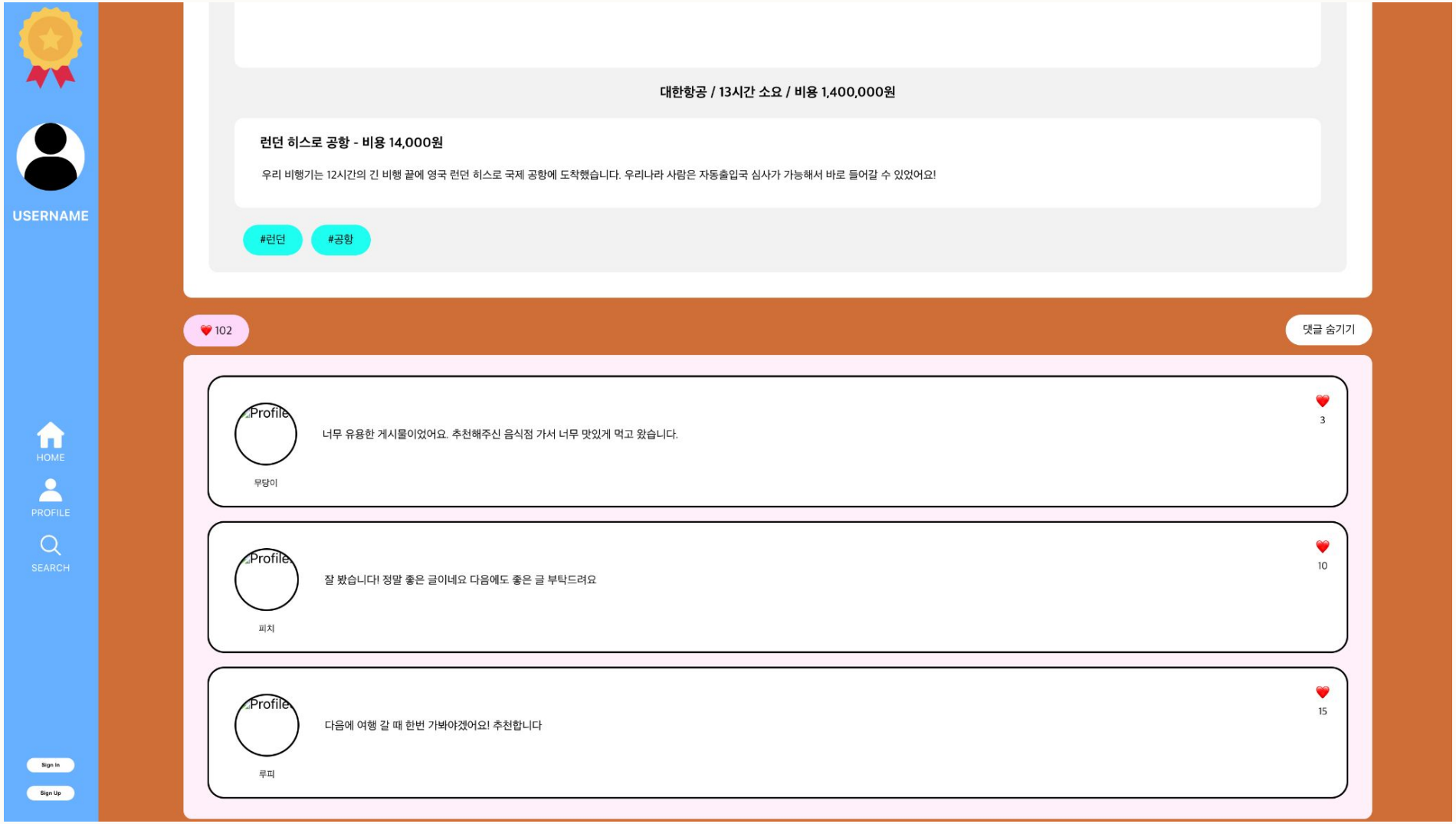
3 페이지 구성도

개인 블로그 포스트 페이지



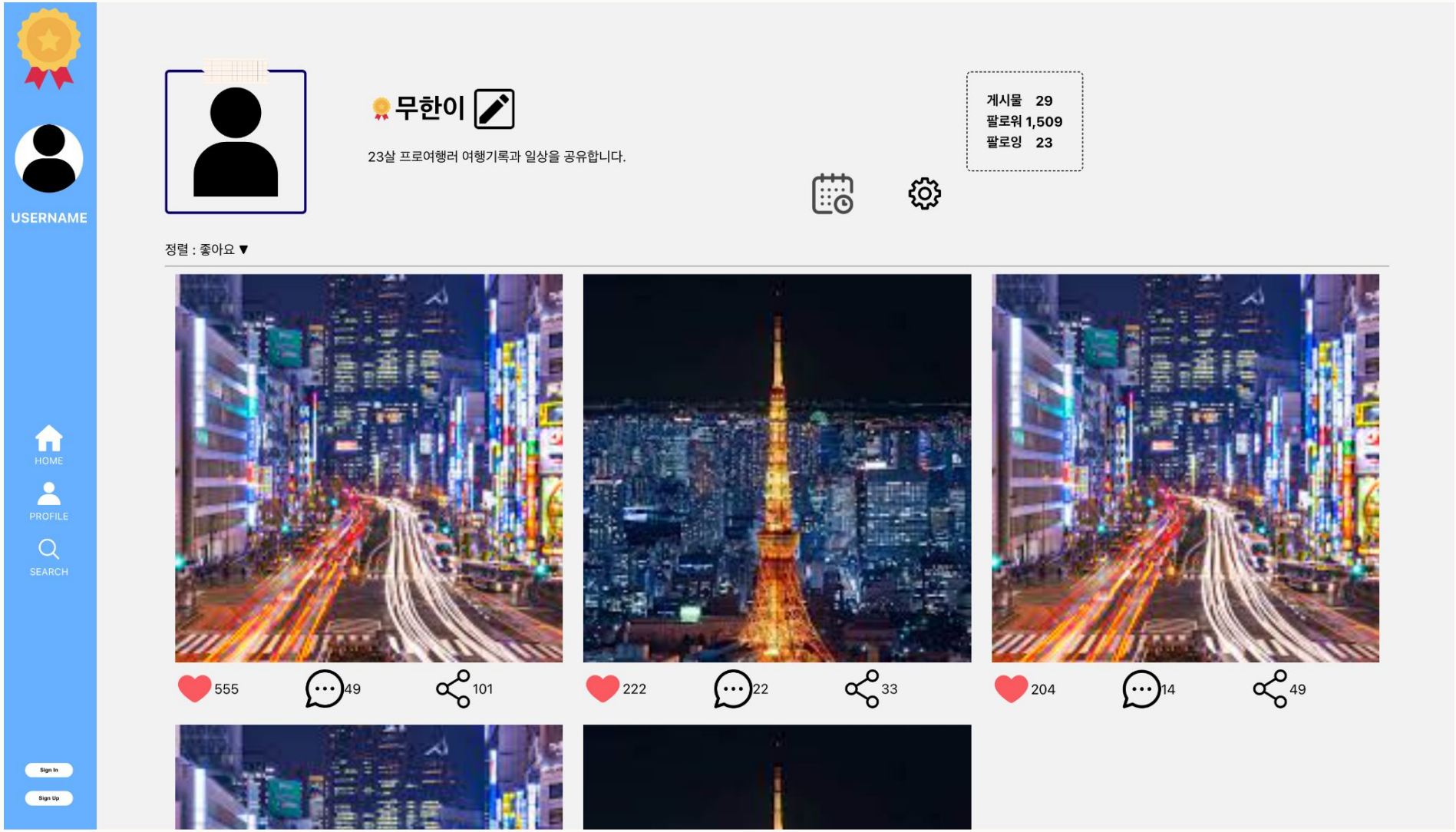
3 페이지 구성도

개인 블로그 포스트 페이지



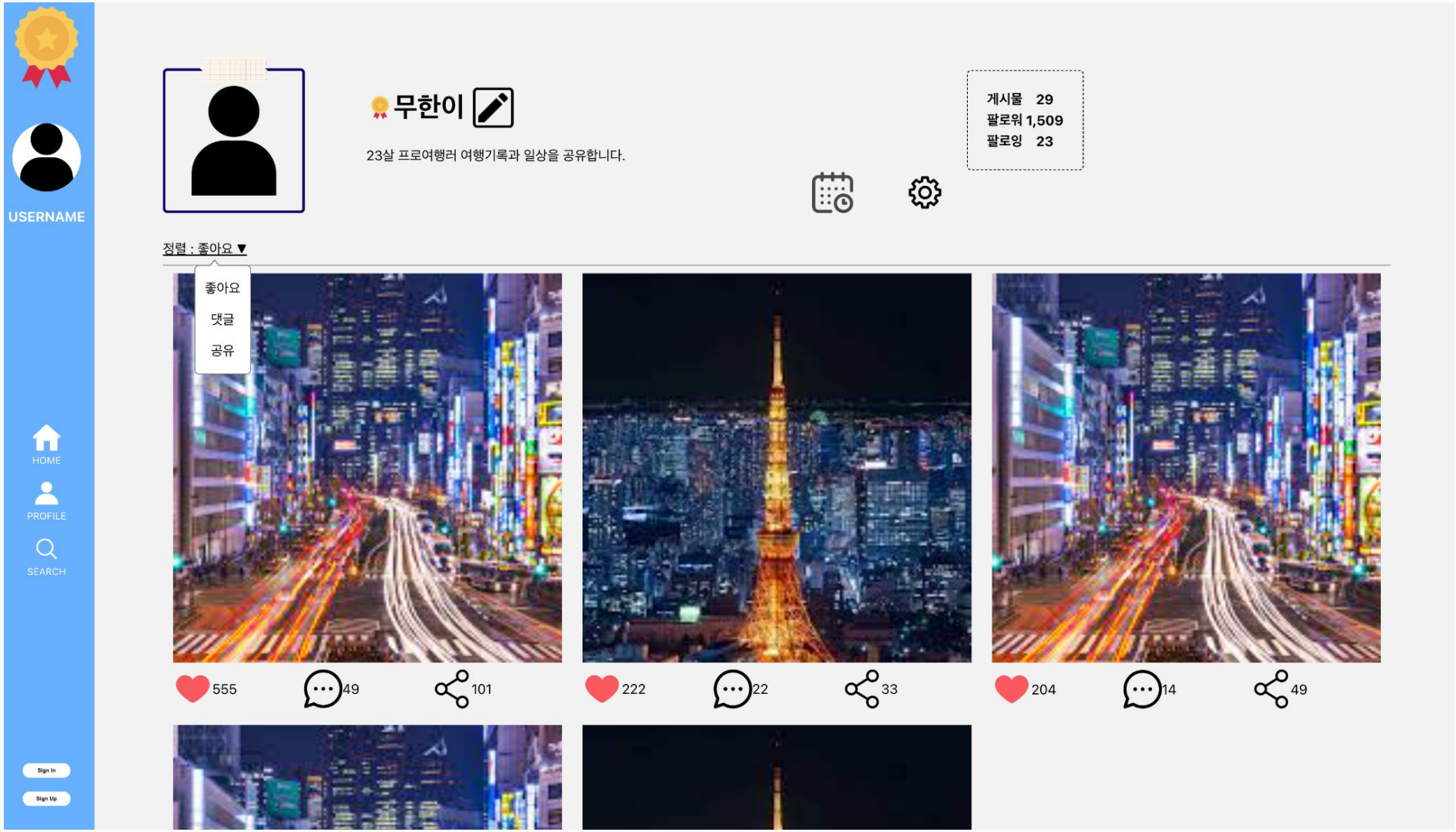
3 페이지 구성도

개인 프로필 페이지



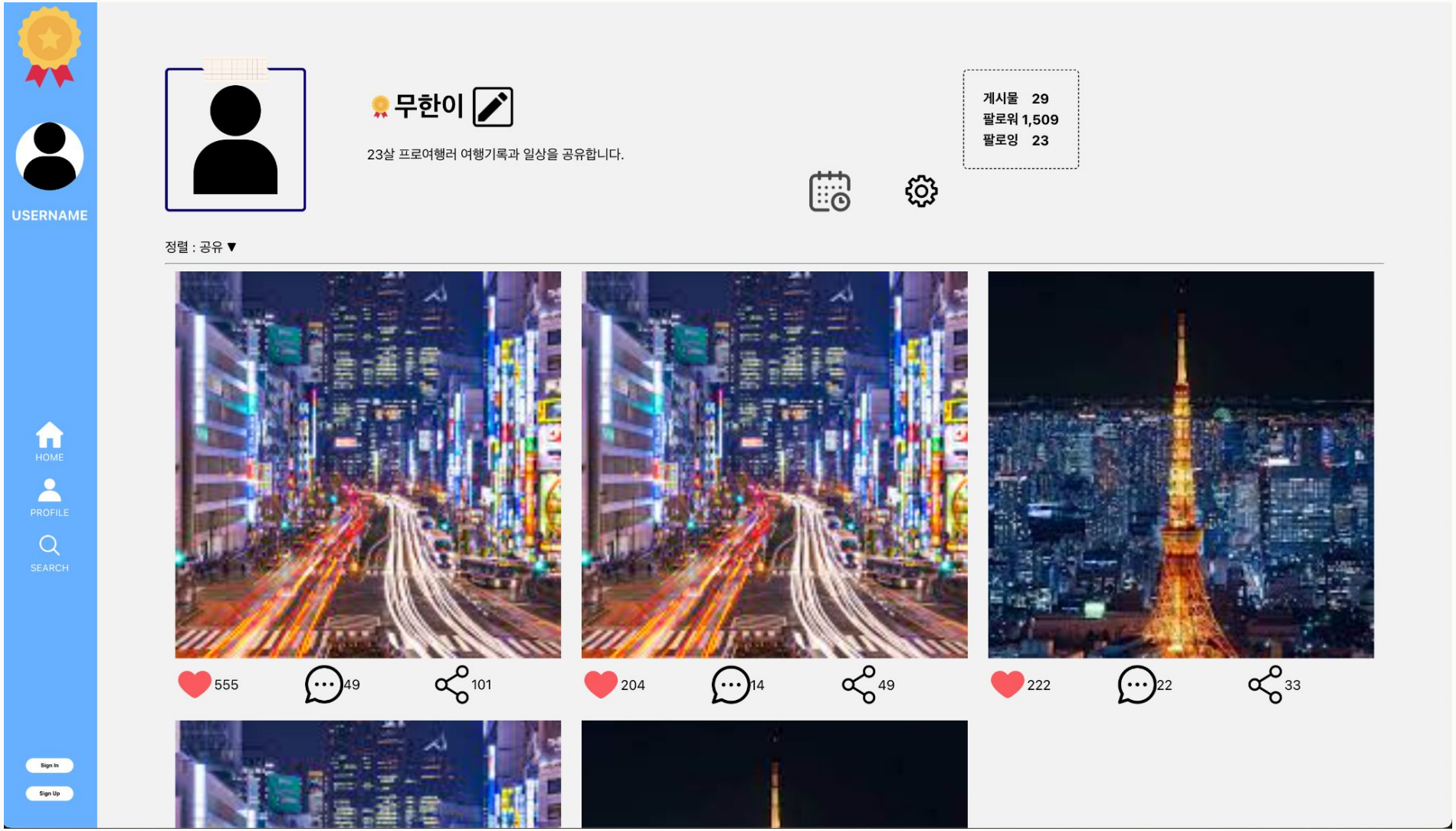
3 페이지 구성도

개인 프로필 페이지



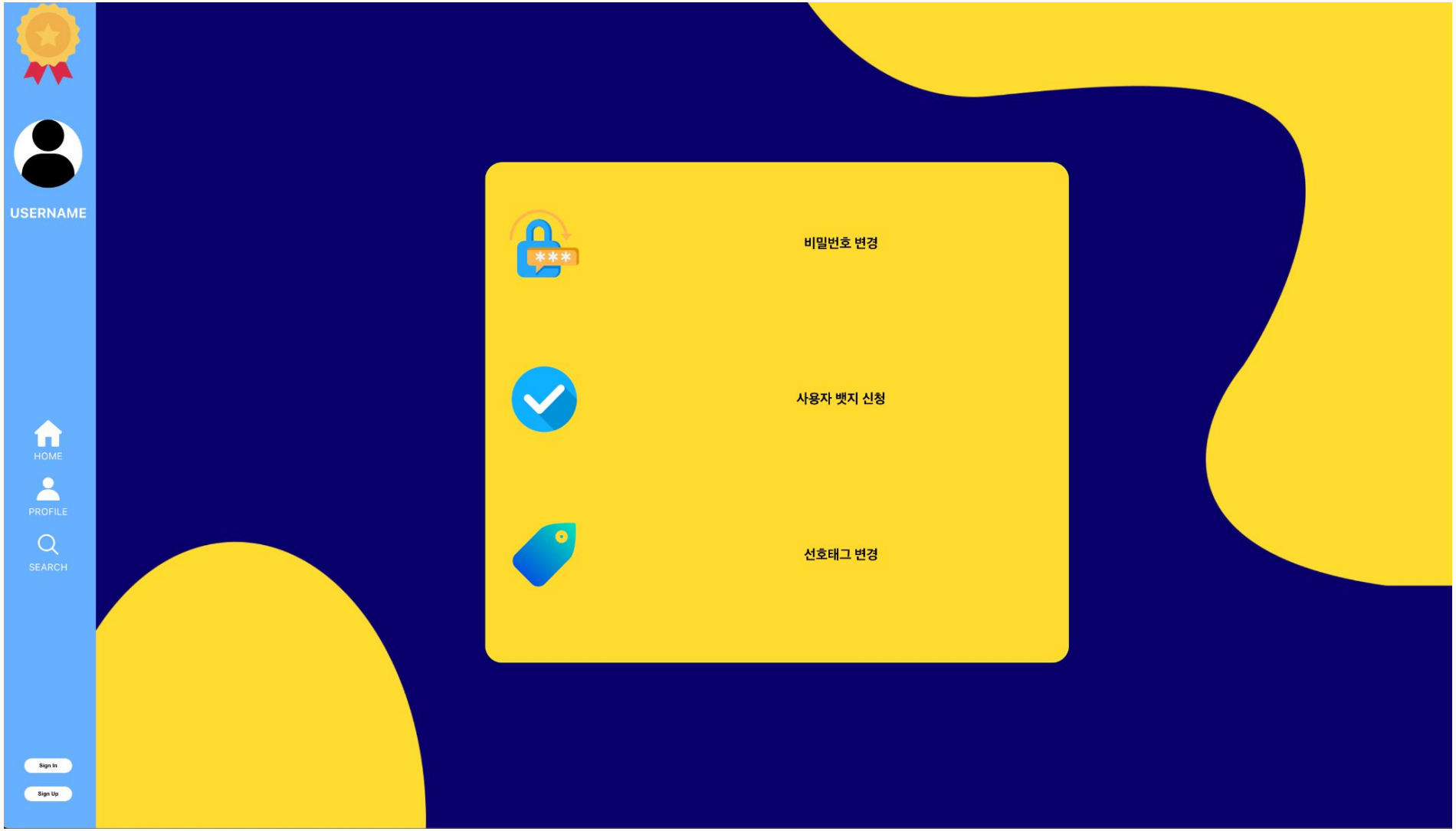
3 페이지 구성도

개인 프로필 페이지



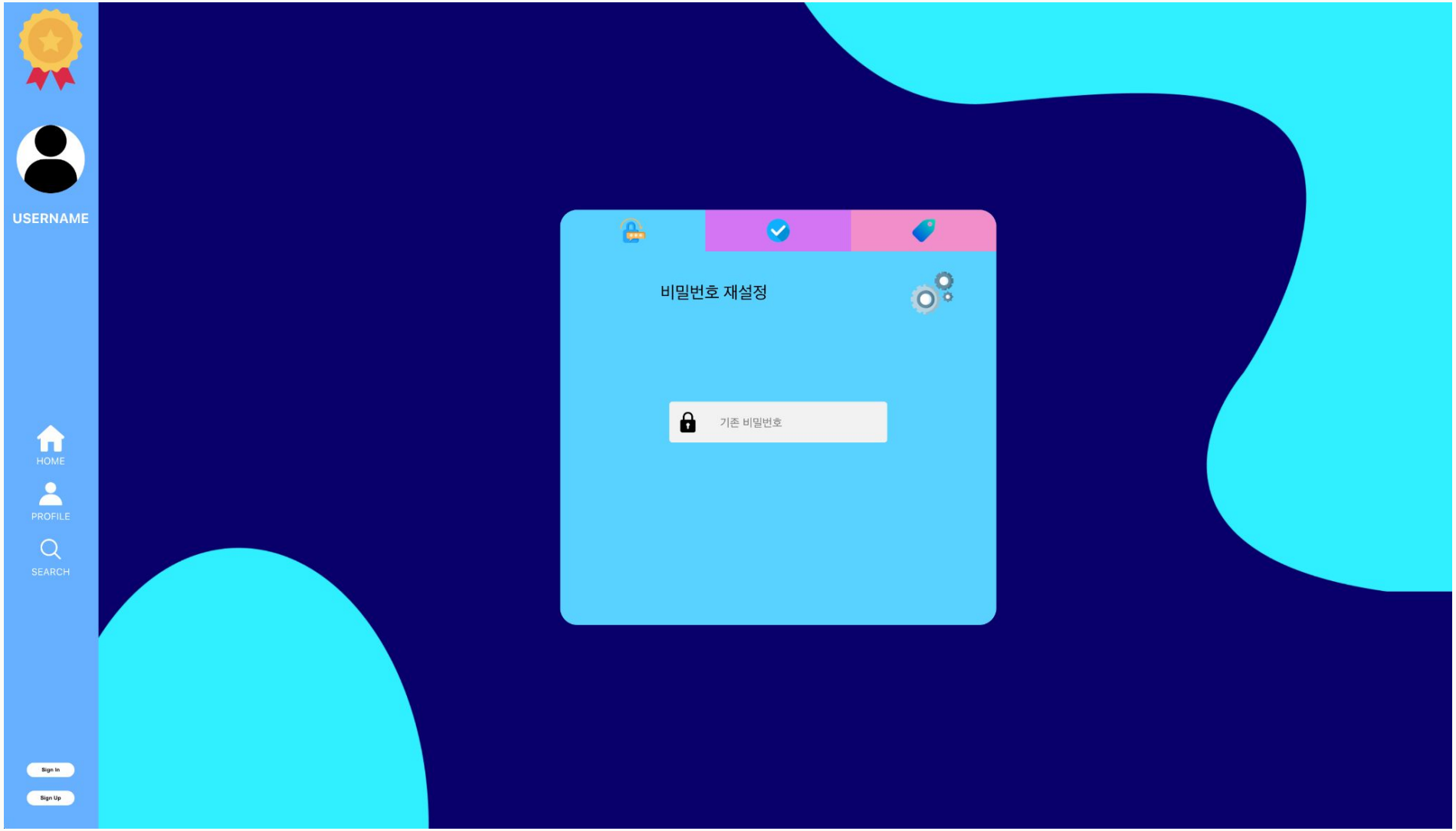
3 페이지 구성도

설정 페이지



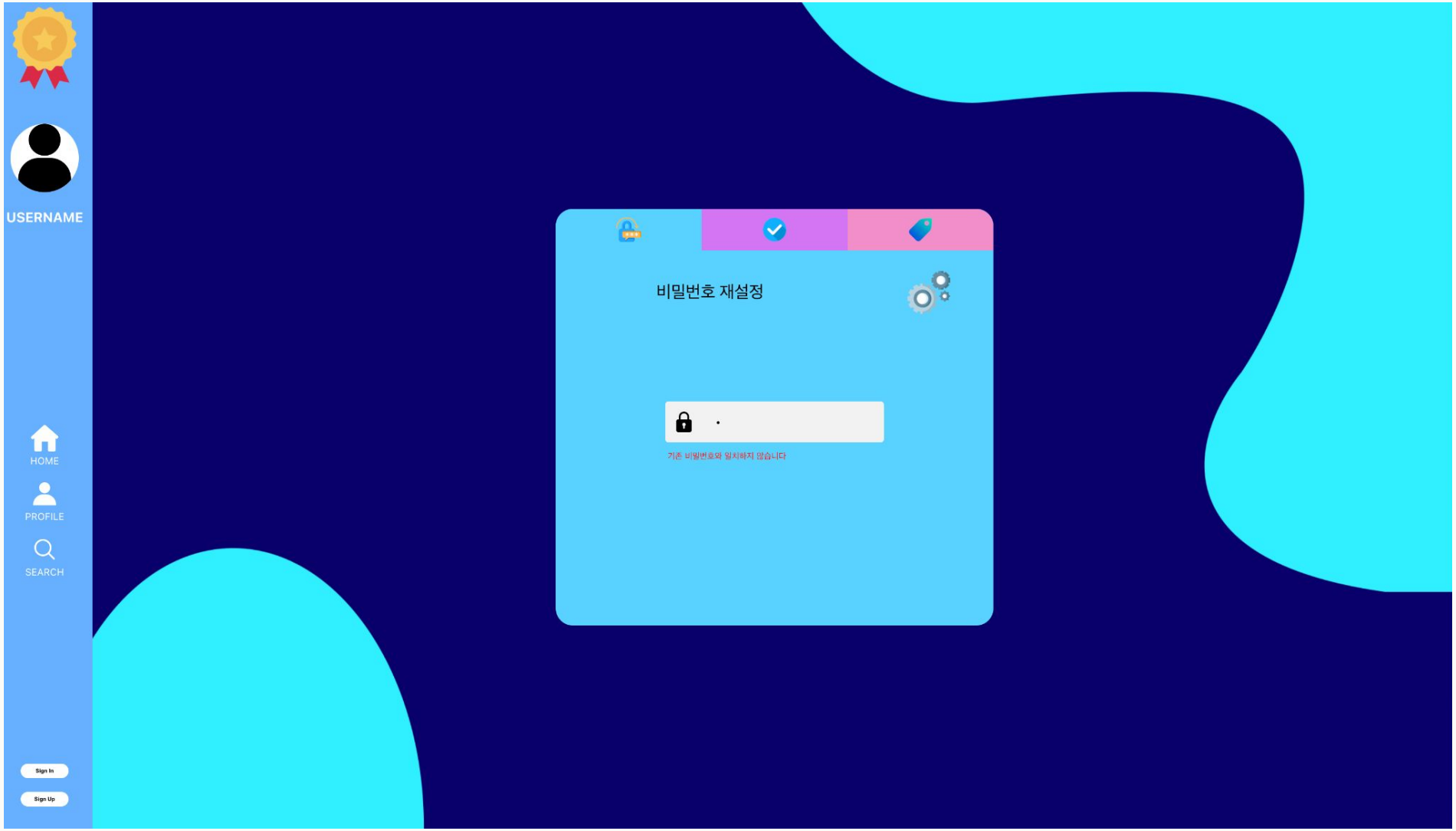
3 페이지 구성도

비밀번호 설정 페이지



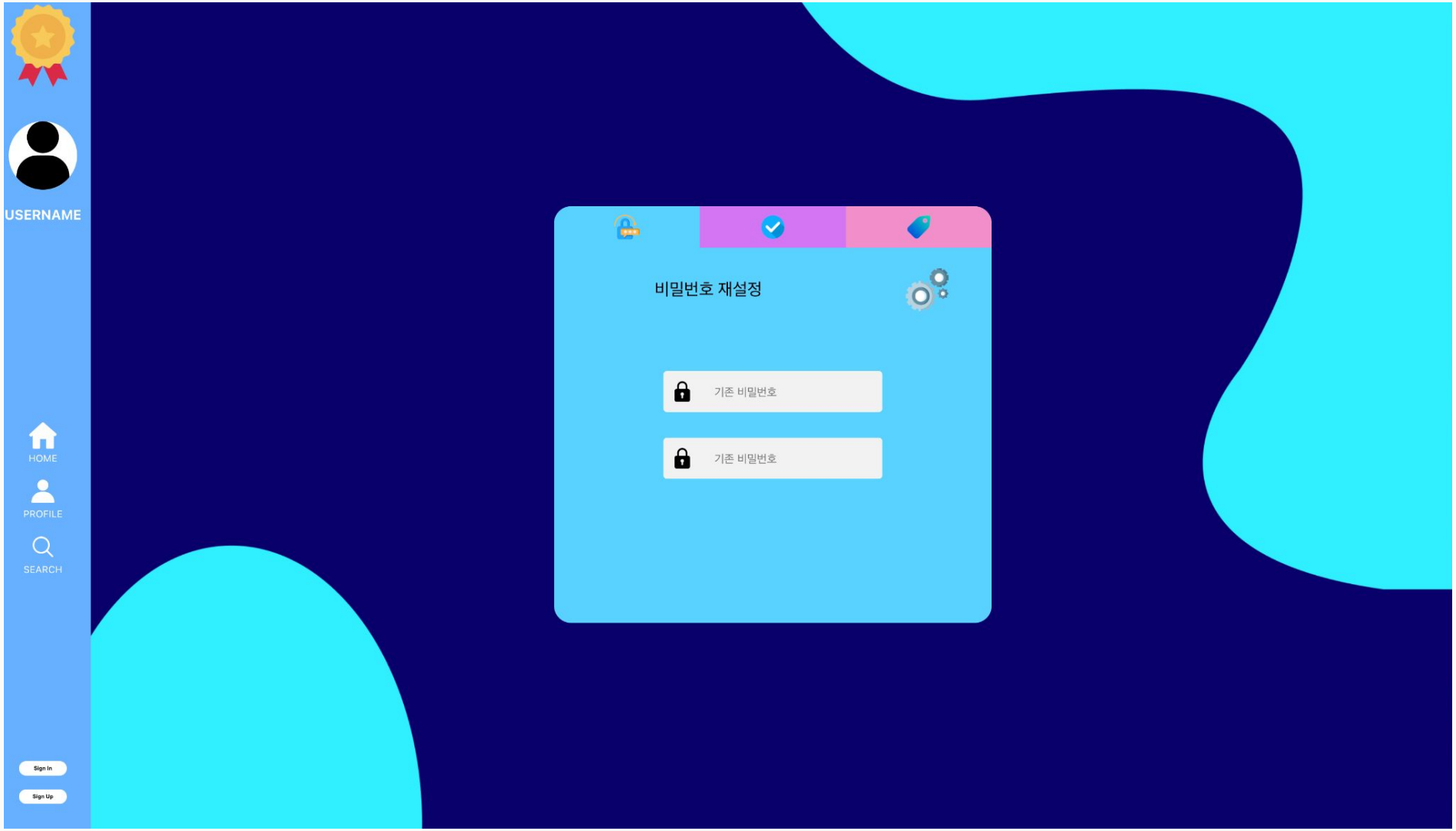
3 페이지 구성도

비밀번호 설정 페이지 - 기존 비밀번호 확인



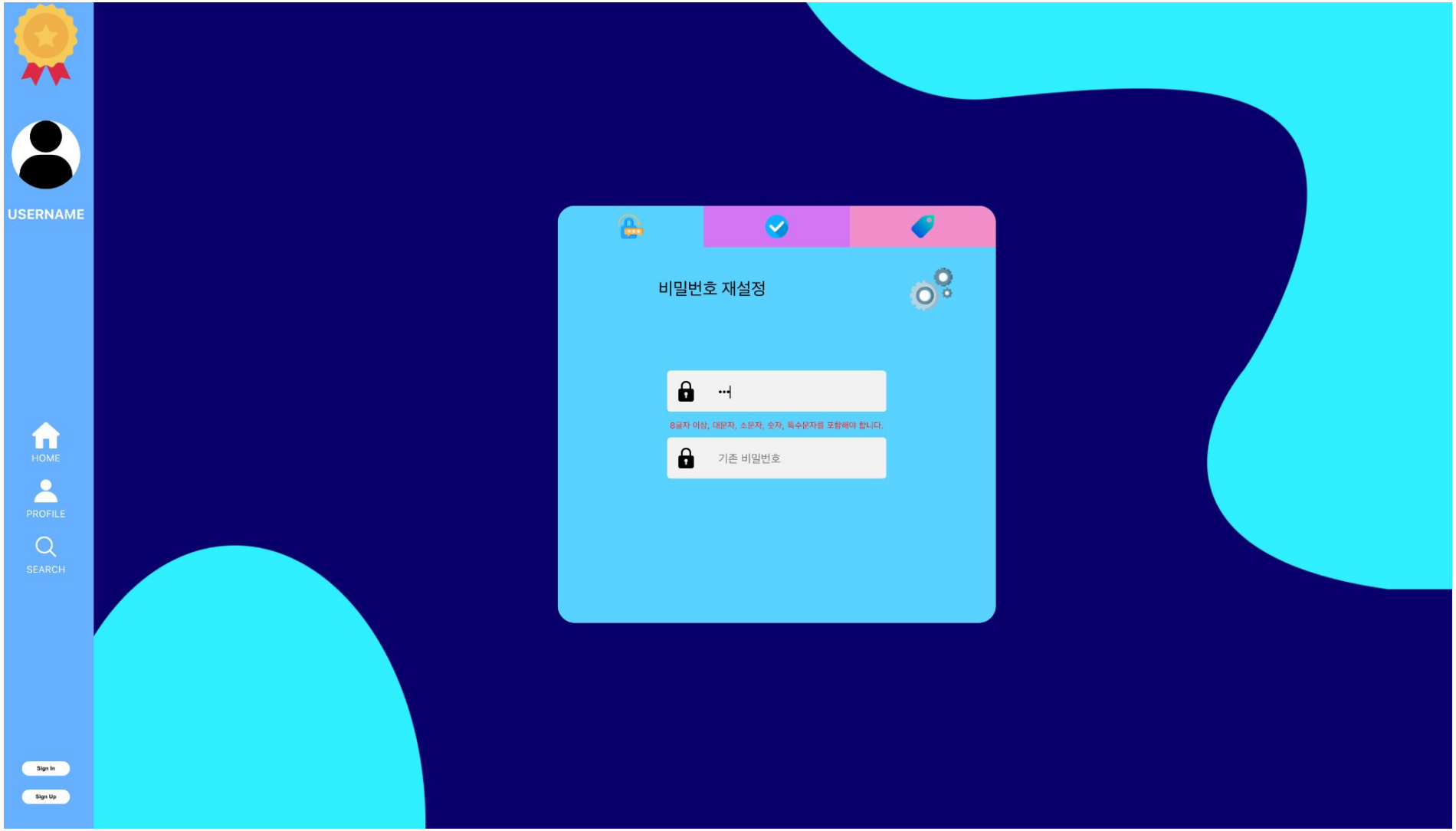
3 페이지 구성도

비밀번호 변경 페이지 - 신규 비밀번호 입력



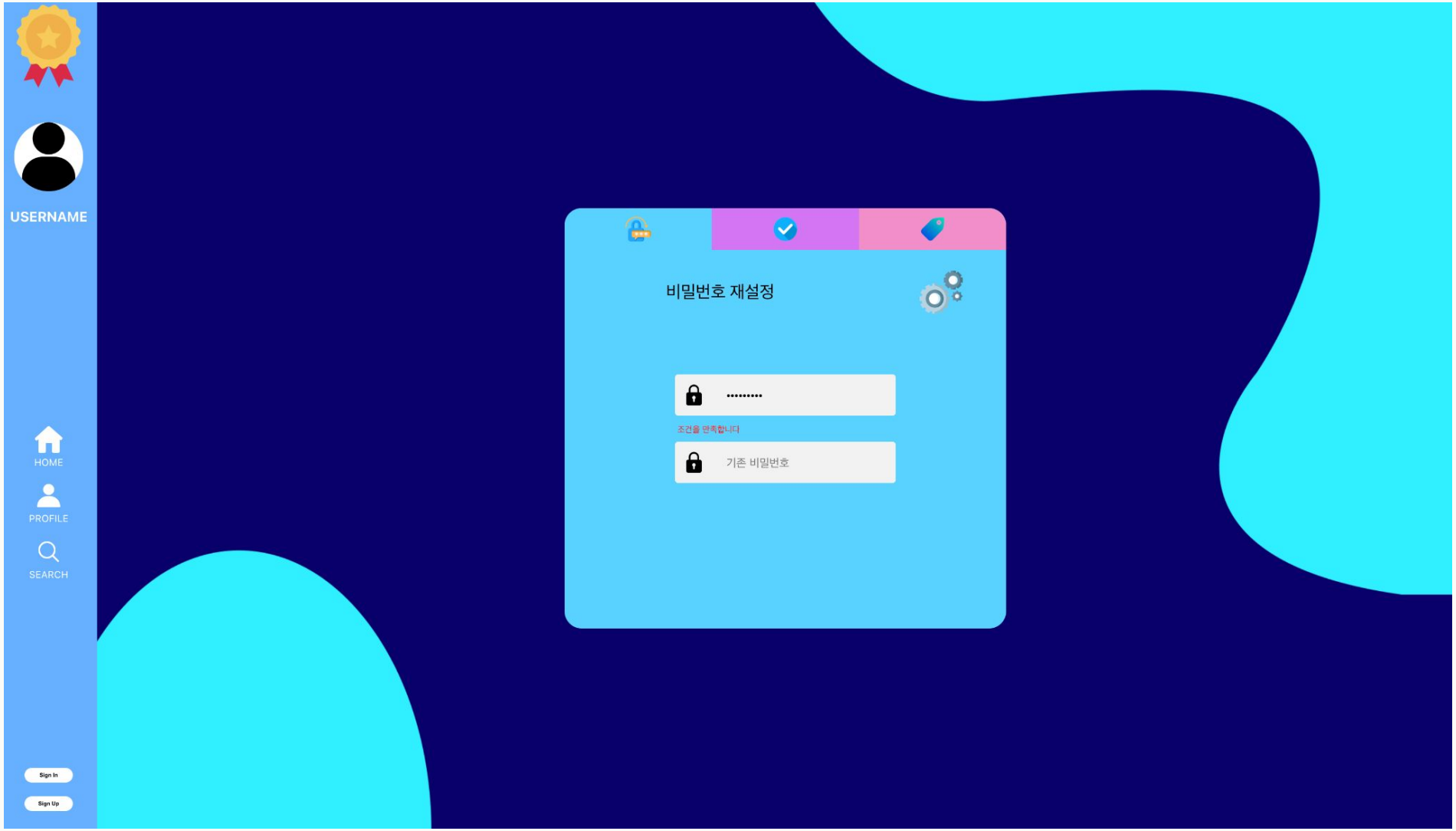
3 페이지 구성도

비밀번호 변경 페이지 - 신규 비밀번호 입력 조건 확인



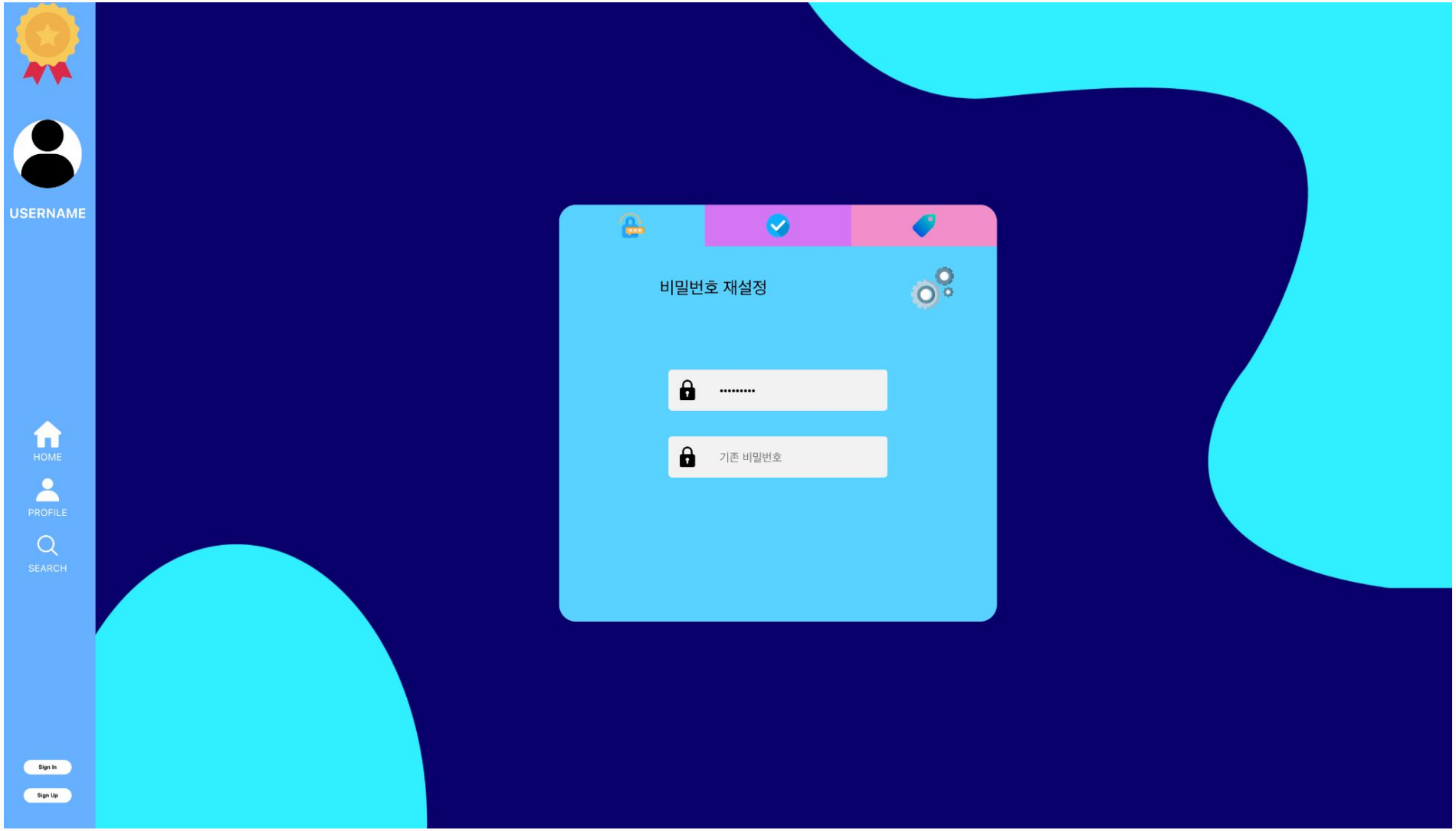
3 페이지 구성도

비밀번호 변경 페이지 - 신규 비밀번호 입력 조건 만족 메시지



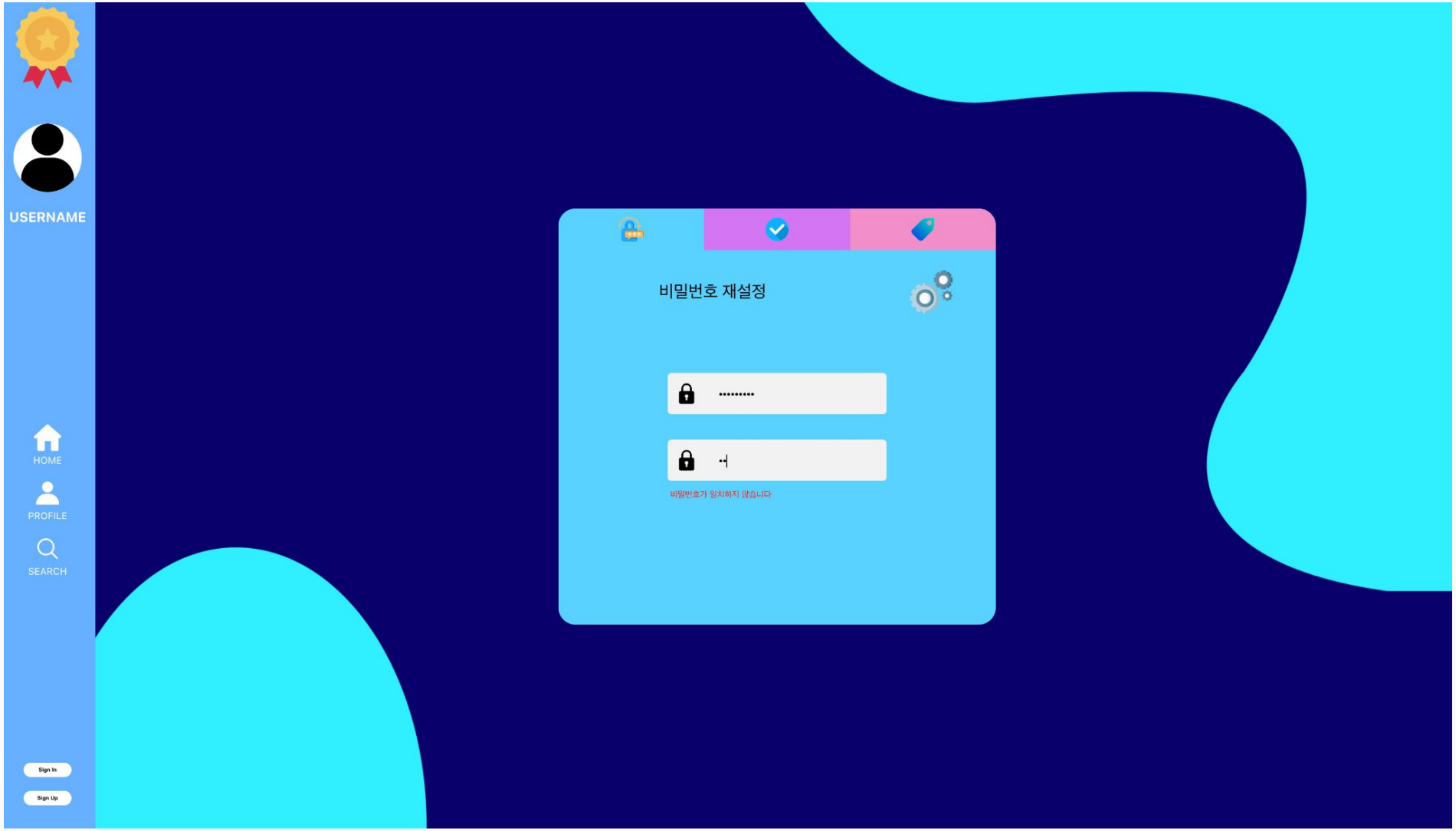
3 페이지

비밀번호 변경 페이지 - 신규 비밀번호 입력 조건 만족 유무



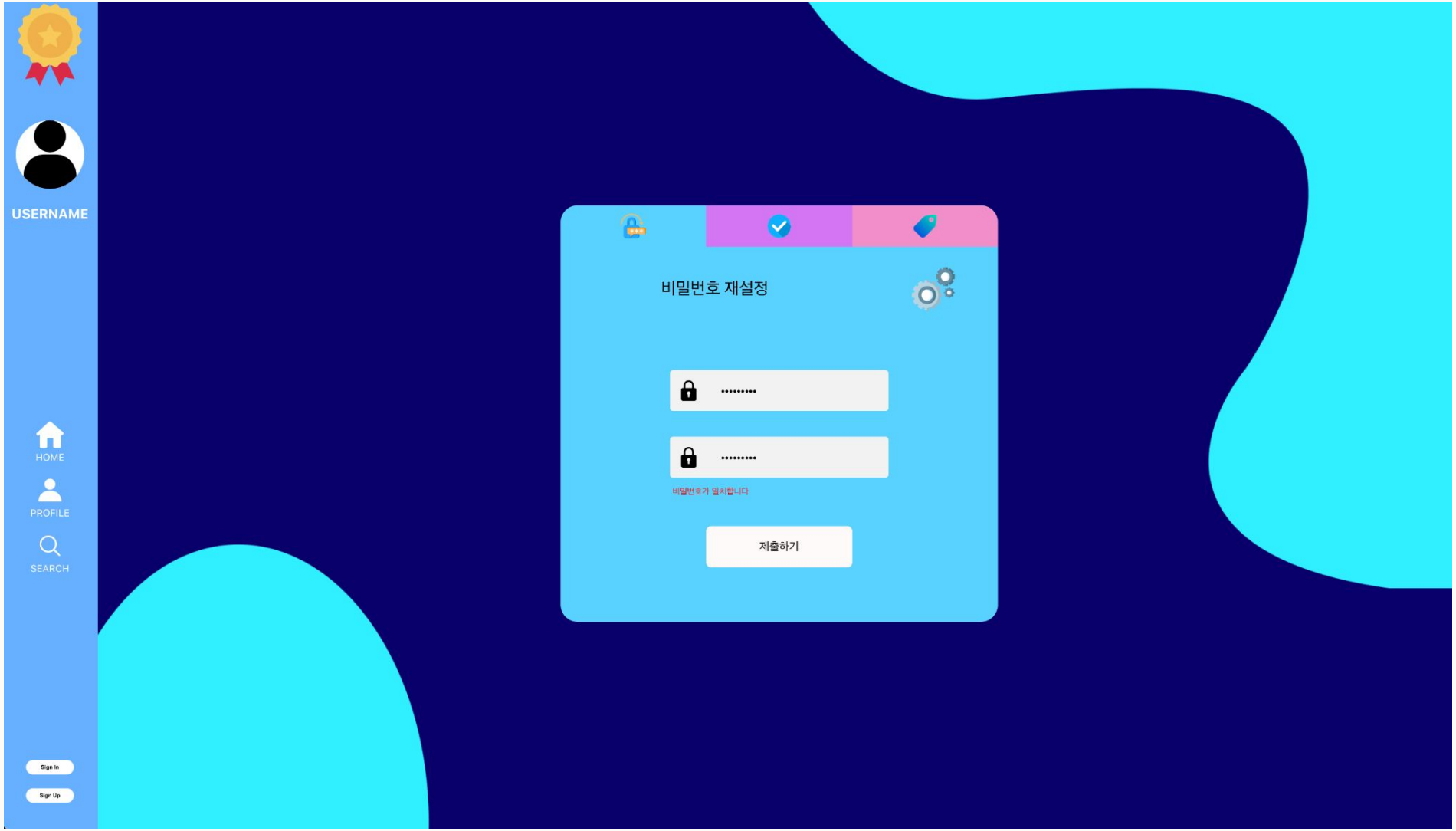
3 페이지 구성도

비밀번호 변경 페이지 - 신규 비밀번호 입력 재입력 확인



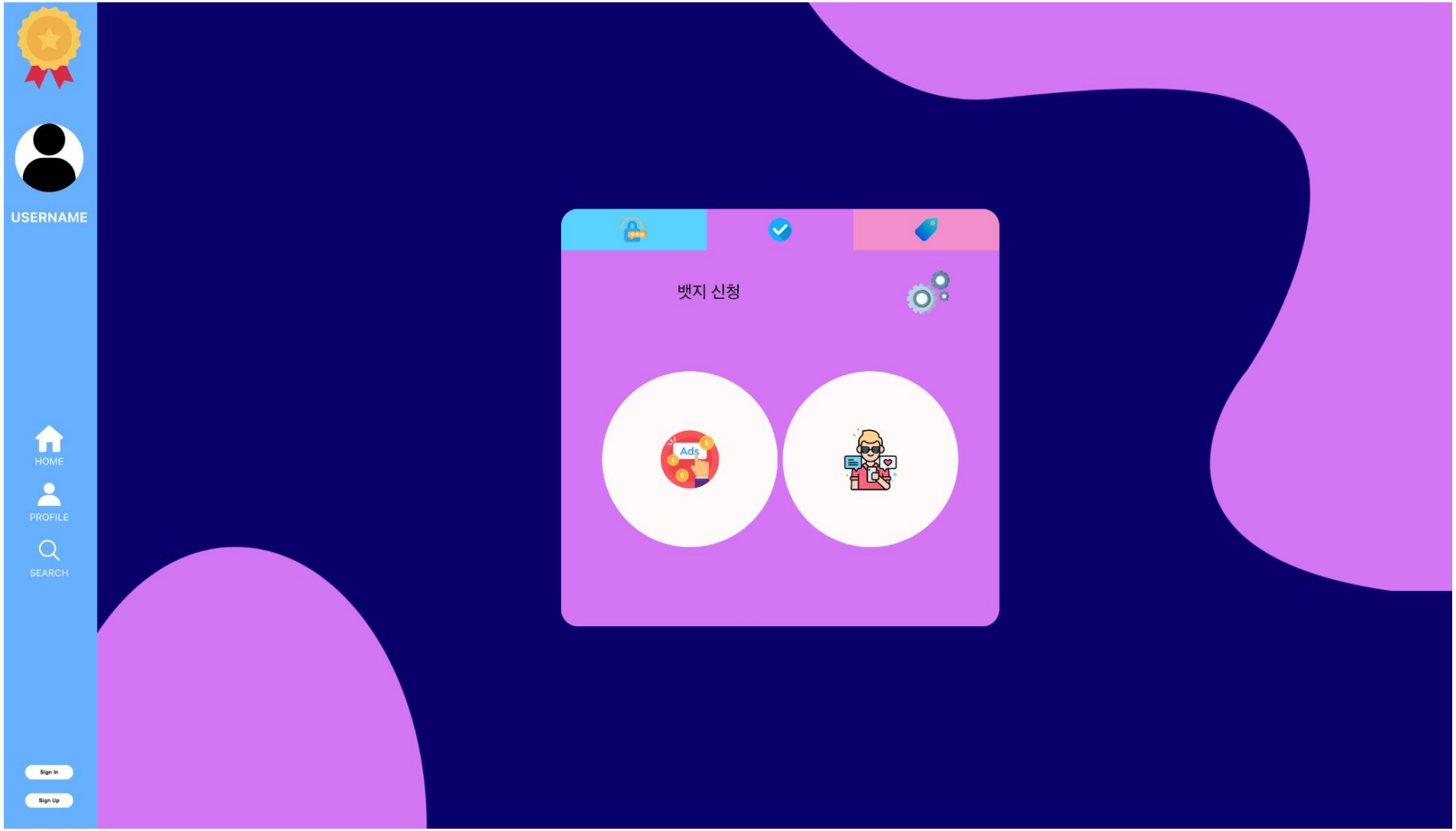
3 페이지 구성도

비밀번호 변경 페이지 - 조건 모두 만족시 입력 버튼 표시



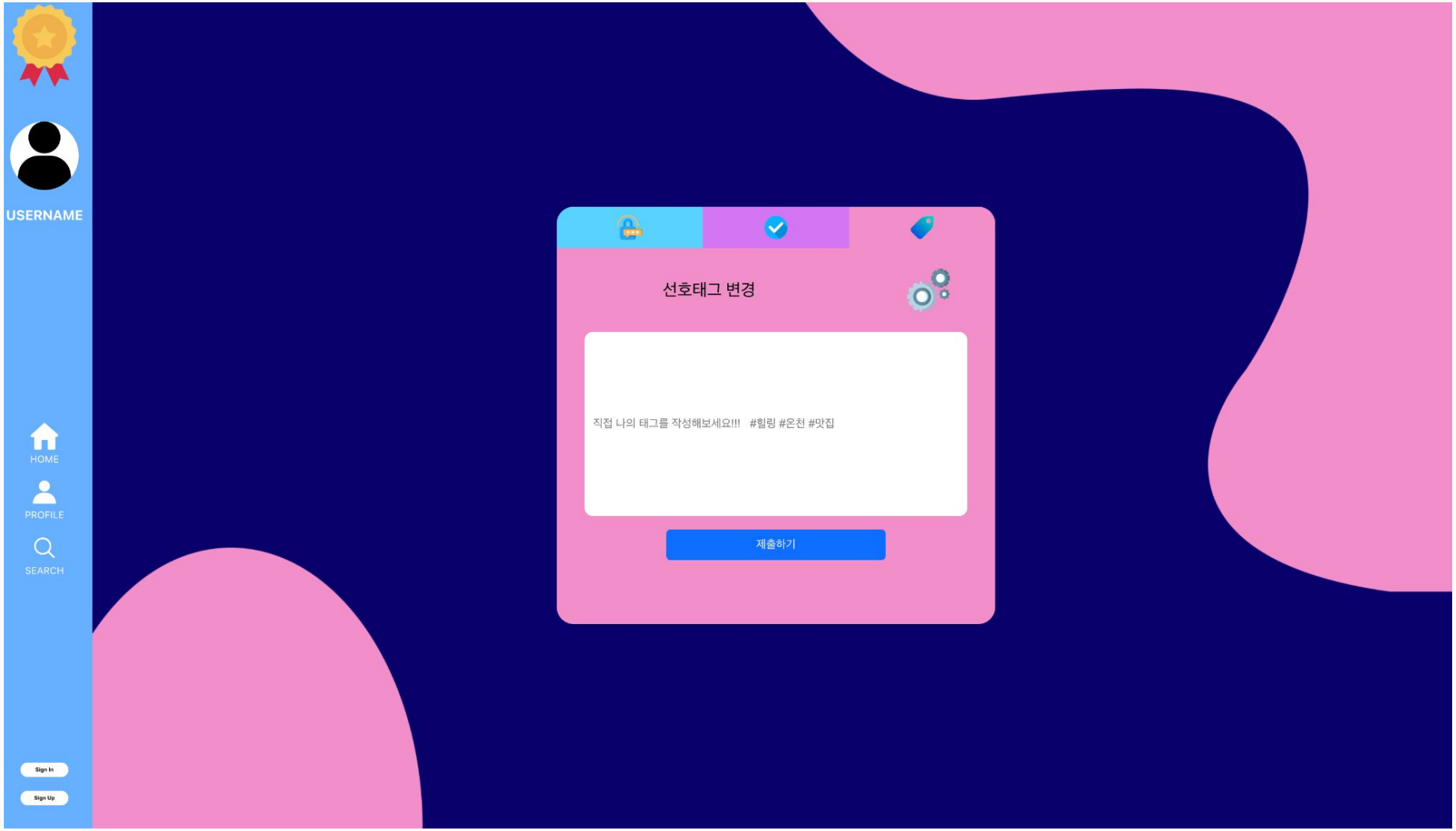
3 페이지 구성도

뱃지 신청 페이지



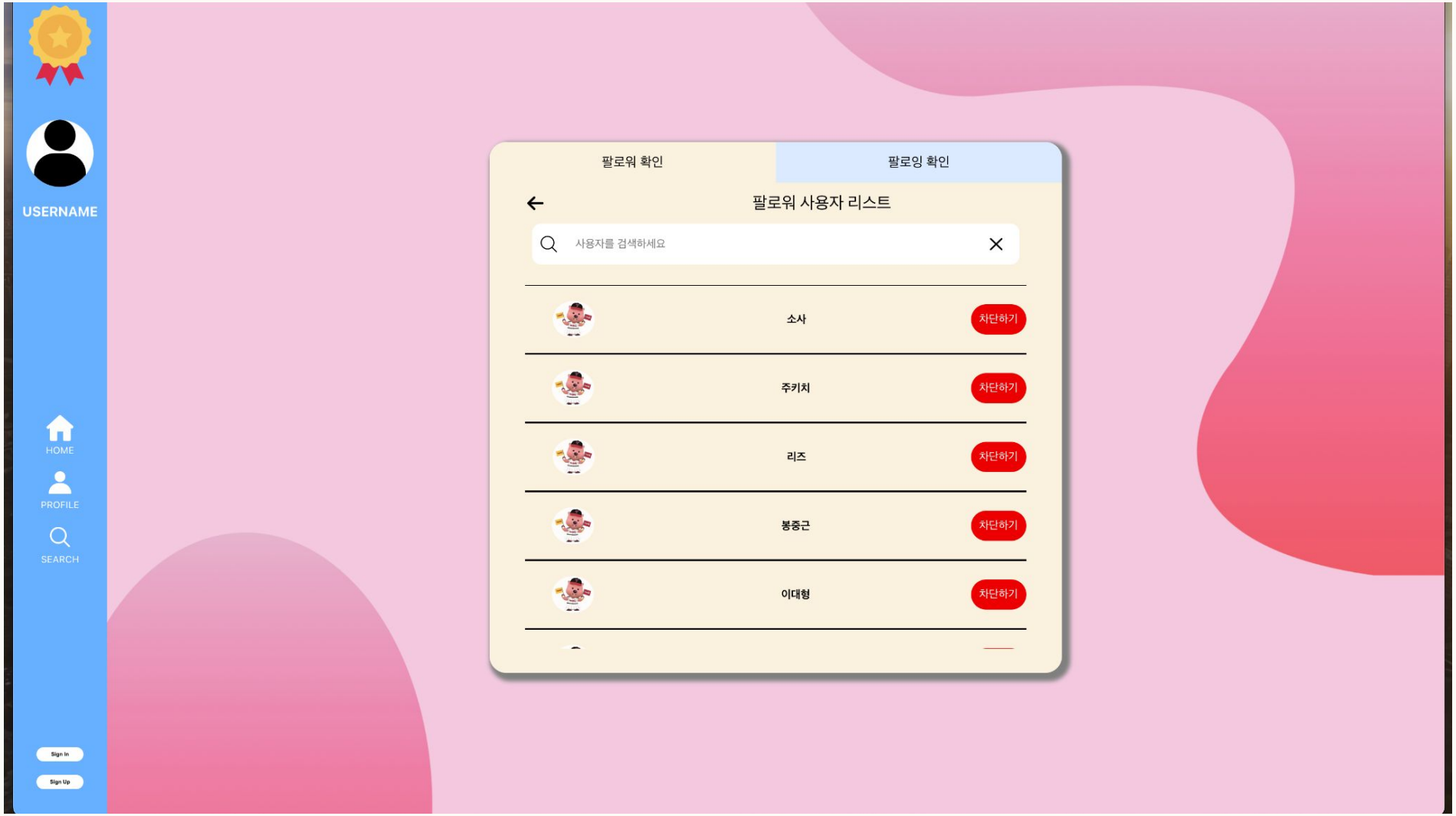
3 페이지 구성도

개인 선호 태그 수정 페이지



3 페이지 구성도

팔로워 확인 페이지

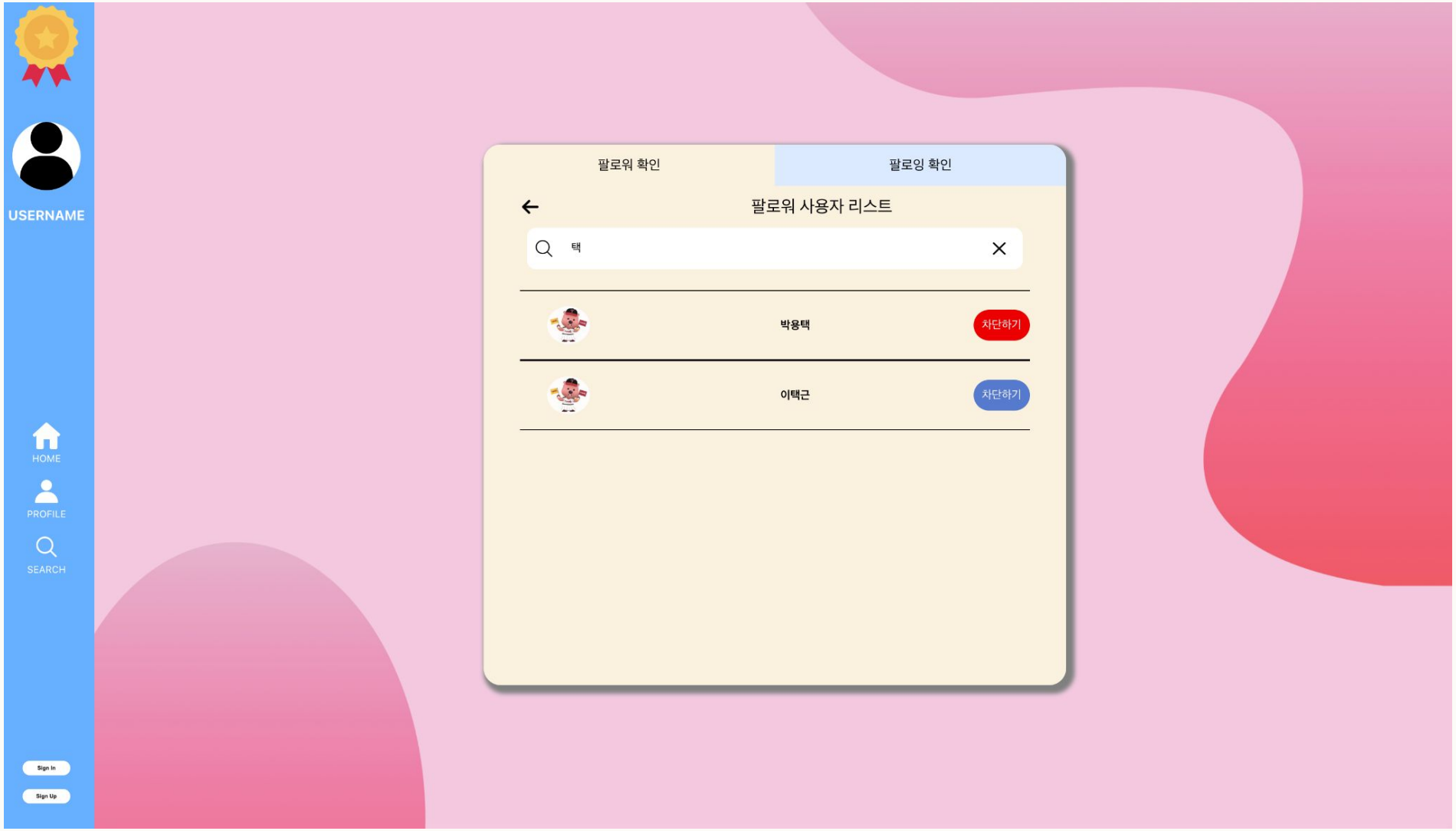


3 페이지 구성도

팔로워 확인 - 사용자 검색 기능

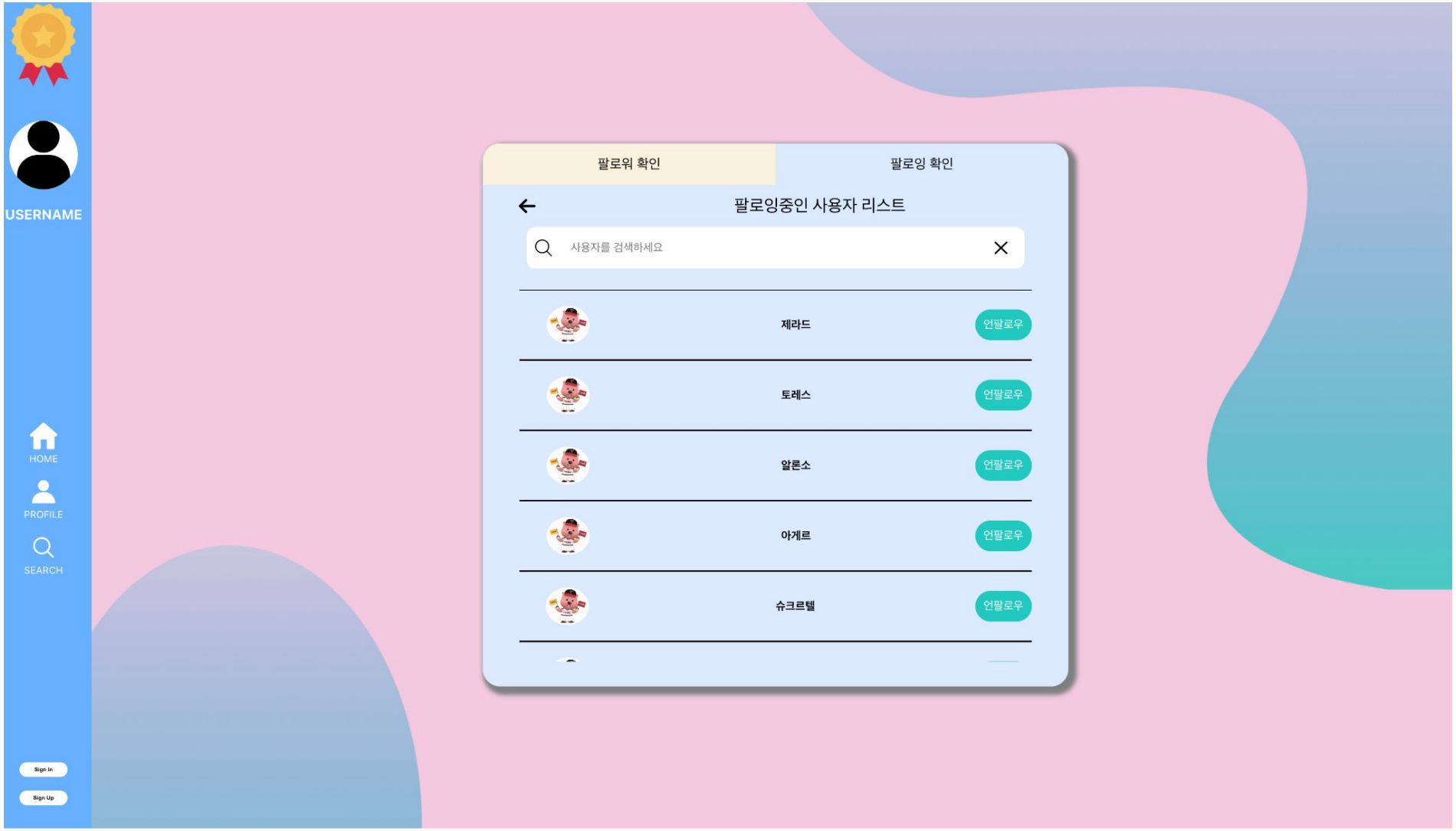
3 페이지 구성도

팔로워 확인 페이지 - 팔로워 삭제



3 페이지 구성도

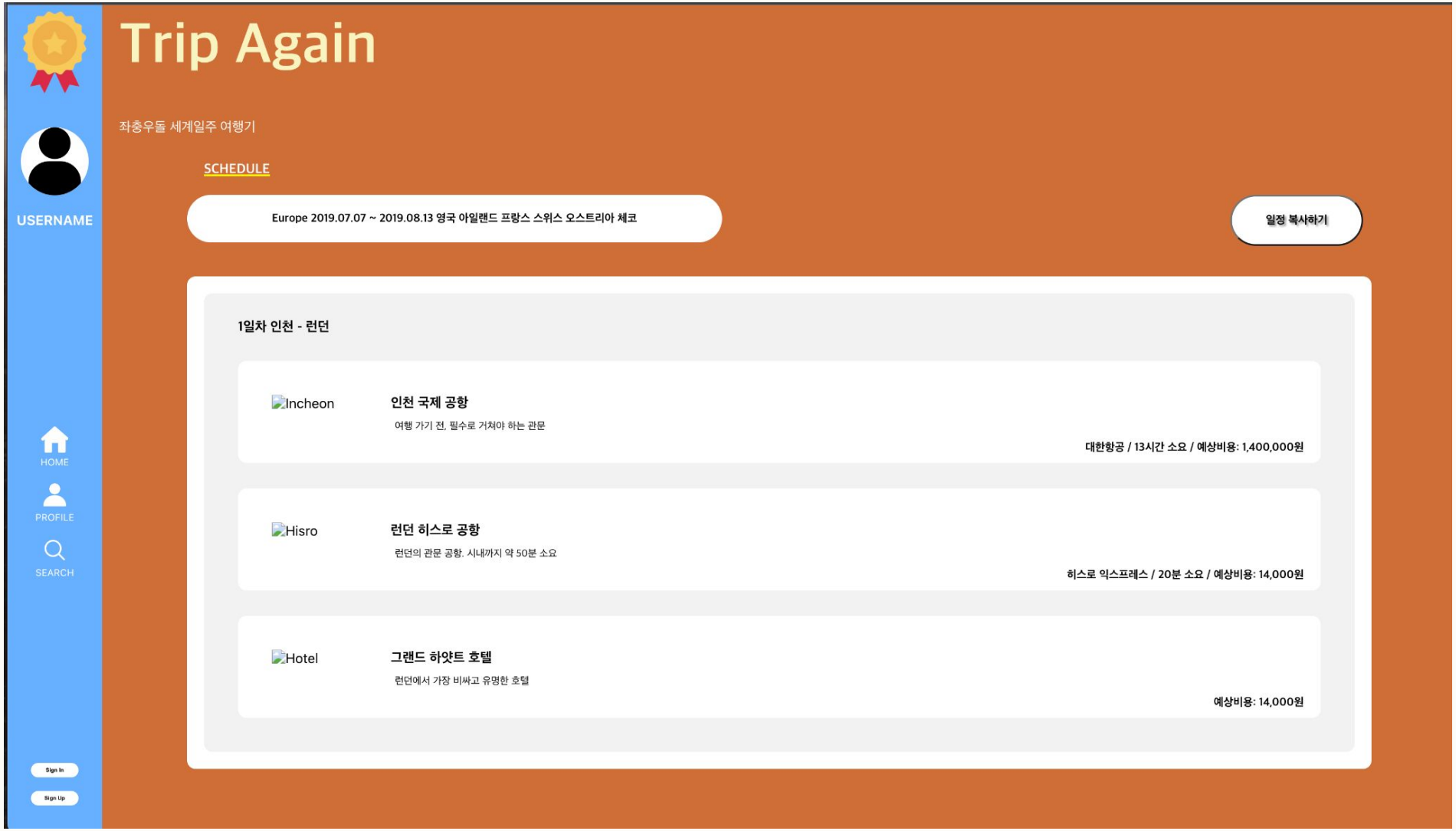
팔로잉 확인 페이지



3 페이지

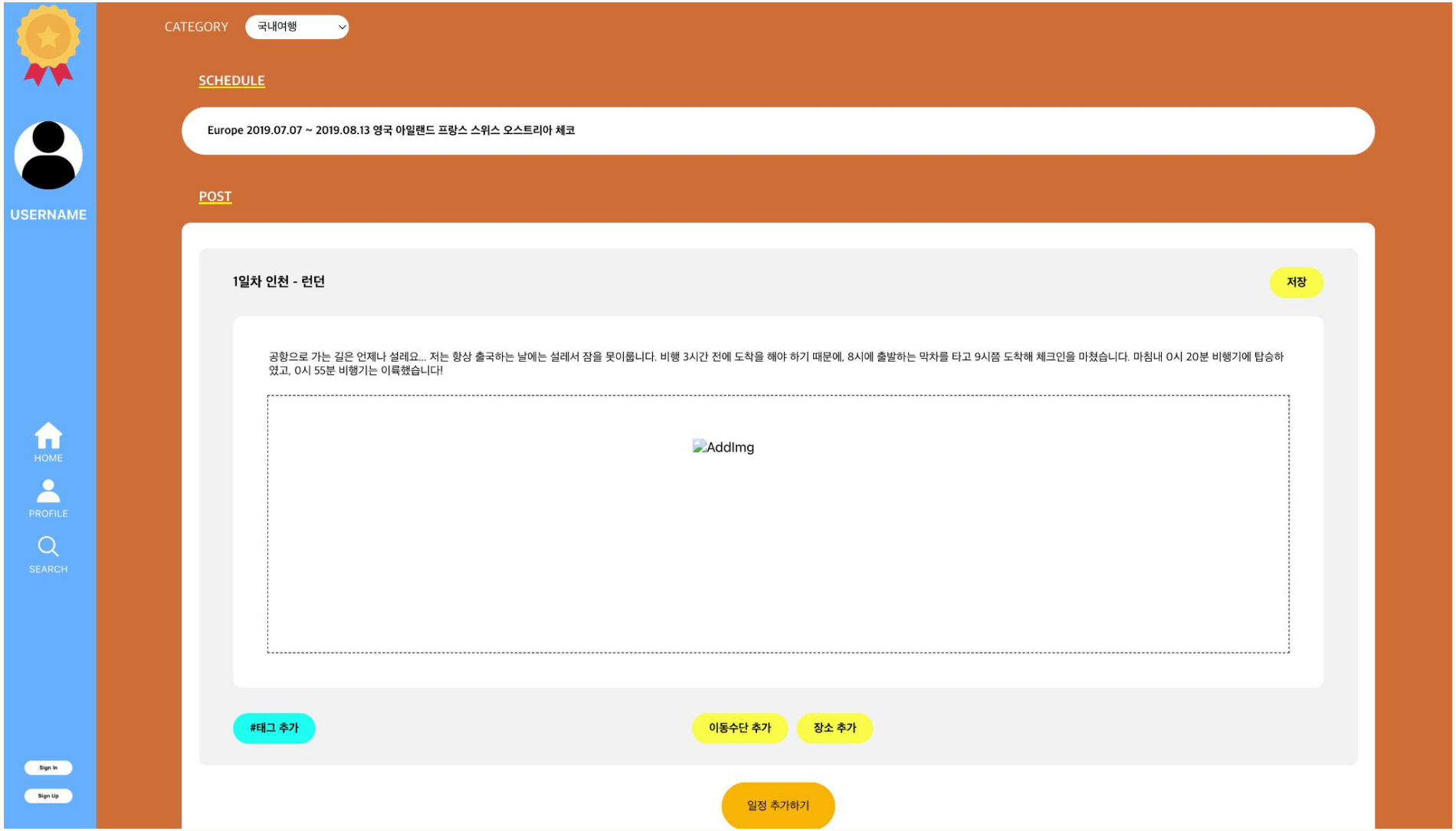
개인 루틴 확인 페이지

구성도



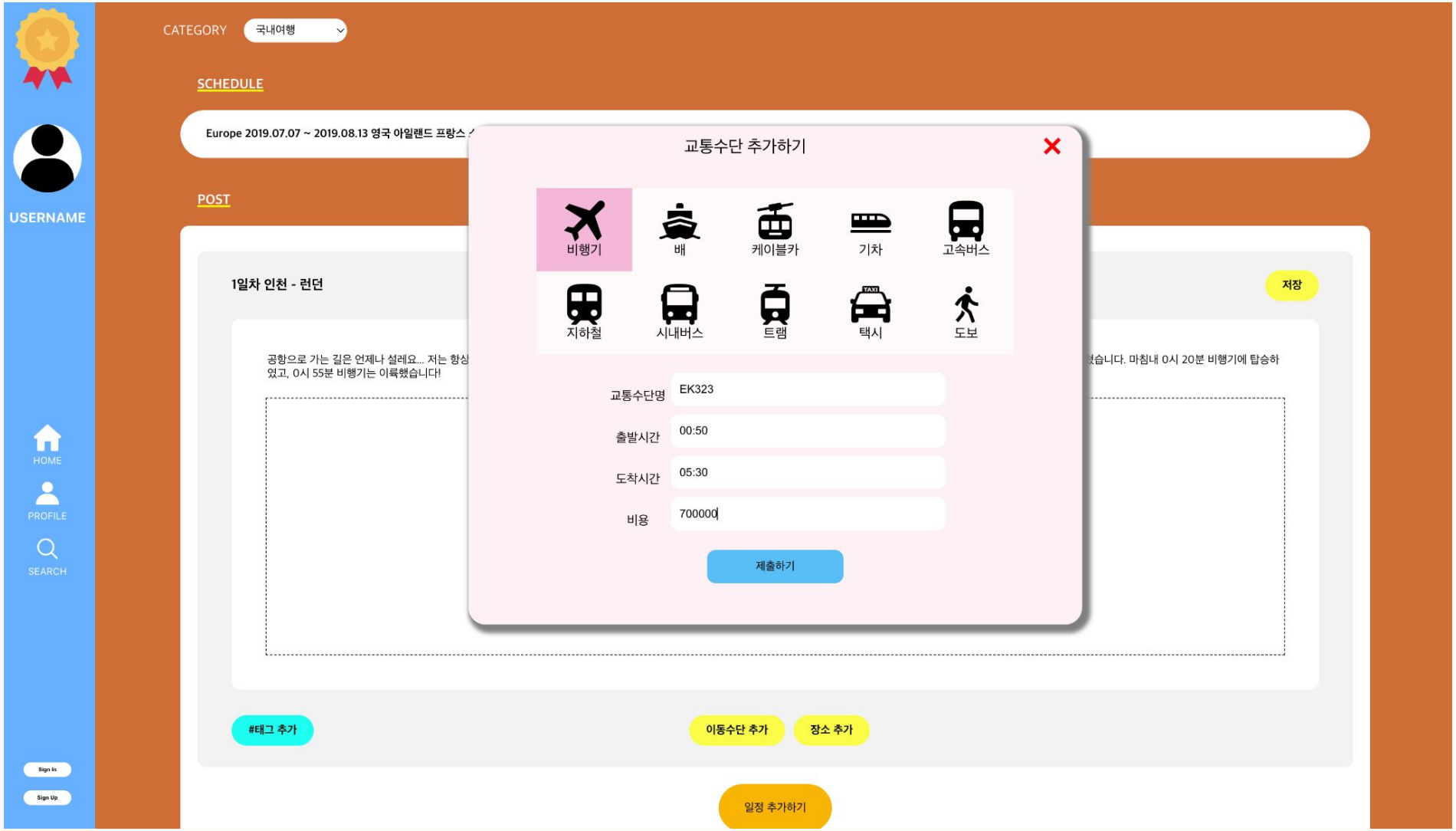
3 페이지

개인 포스트 등록 페이지



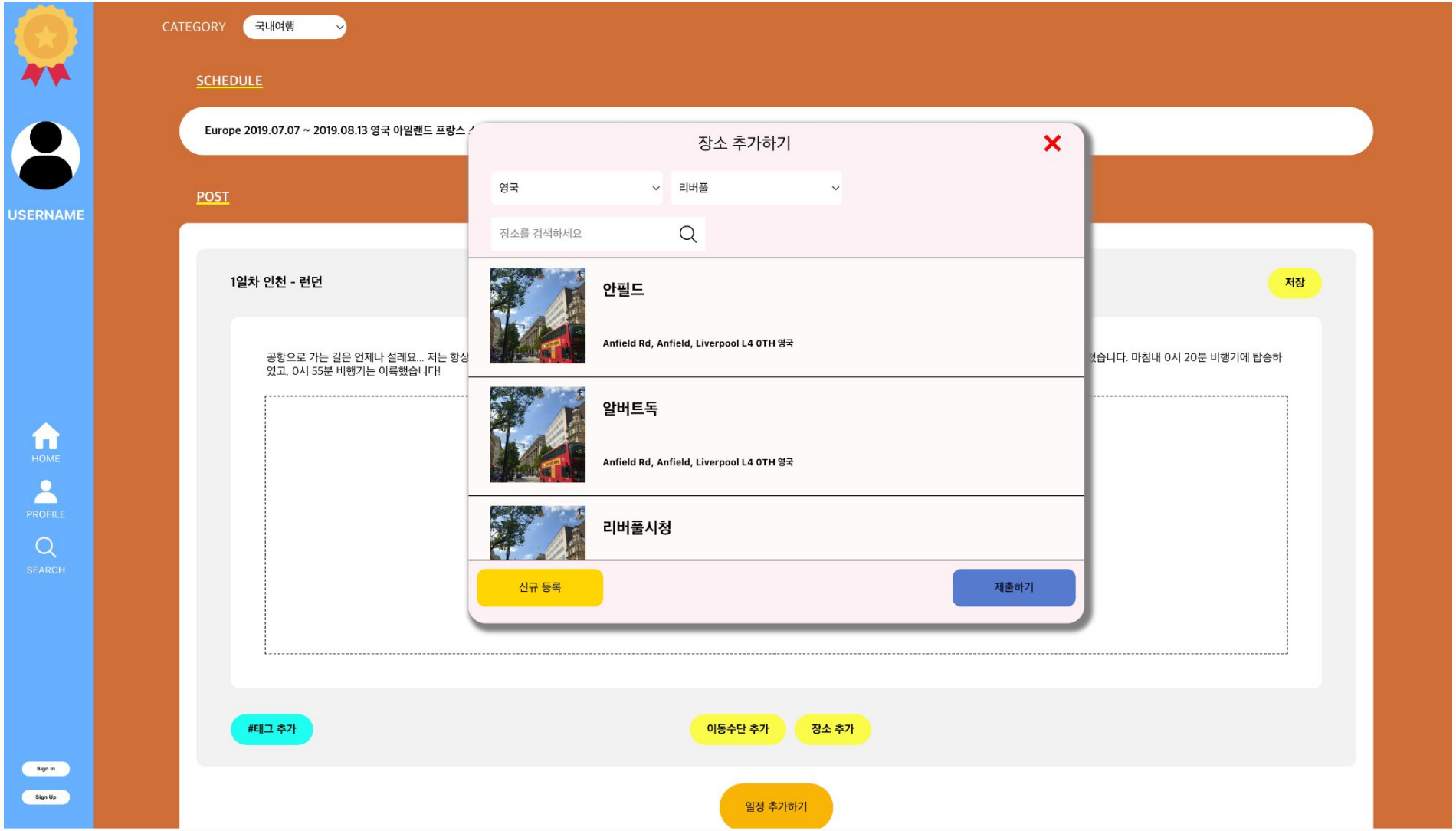
3 페이지 구성도

교통수단 등록 팝업



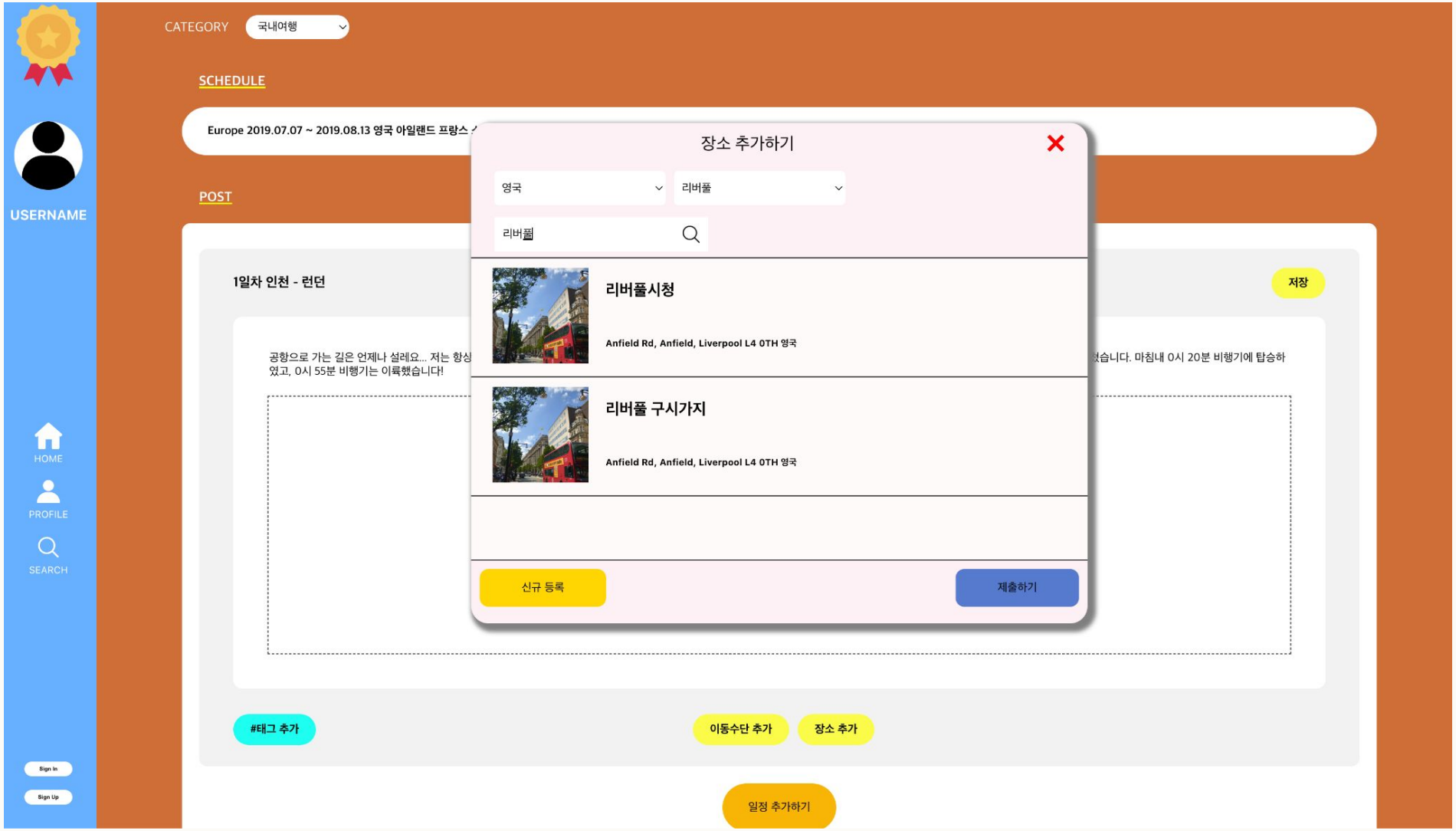
3 페이지 구성도

장소 등록 팝업



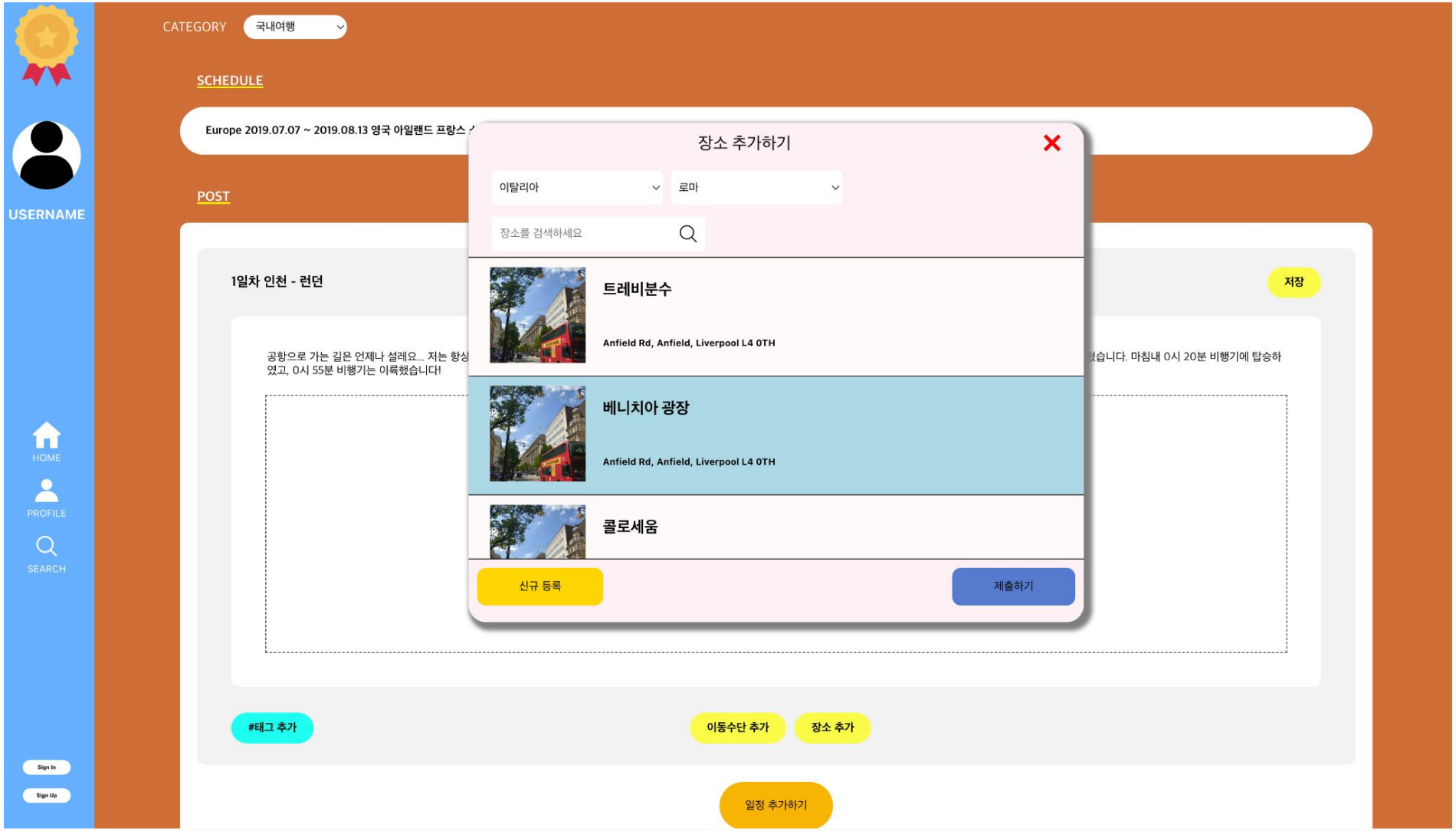
3 페이지 구성도

장소 추가 팝업 - 장소 검색



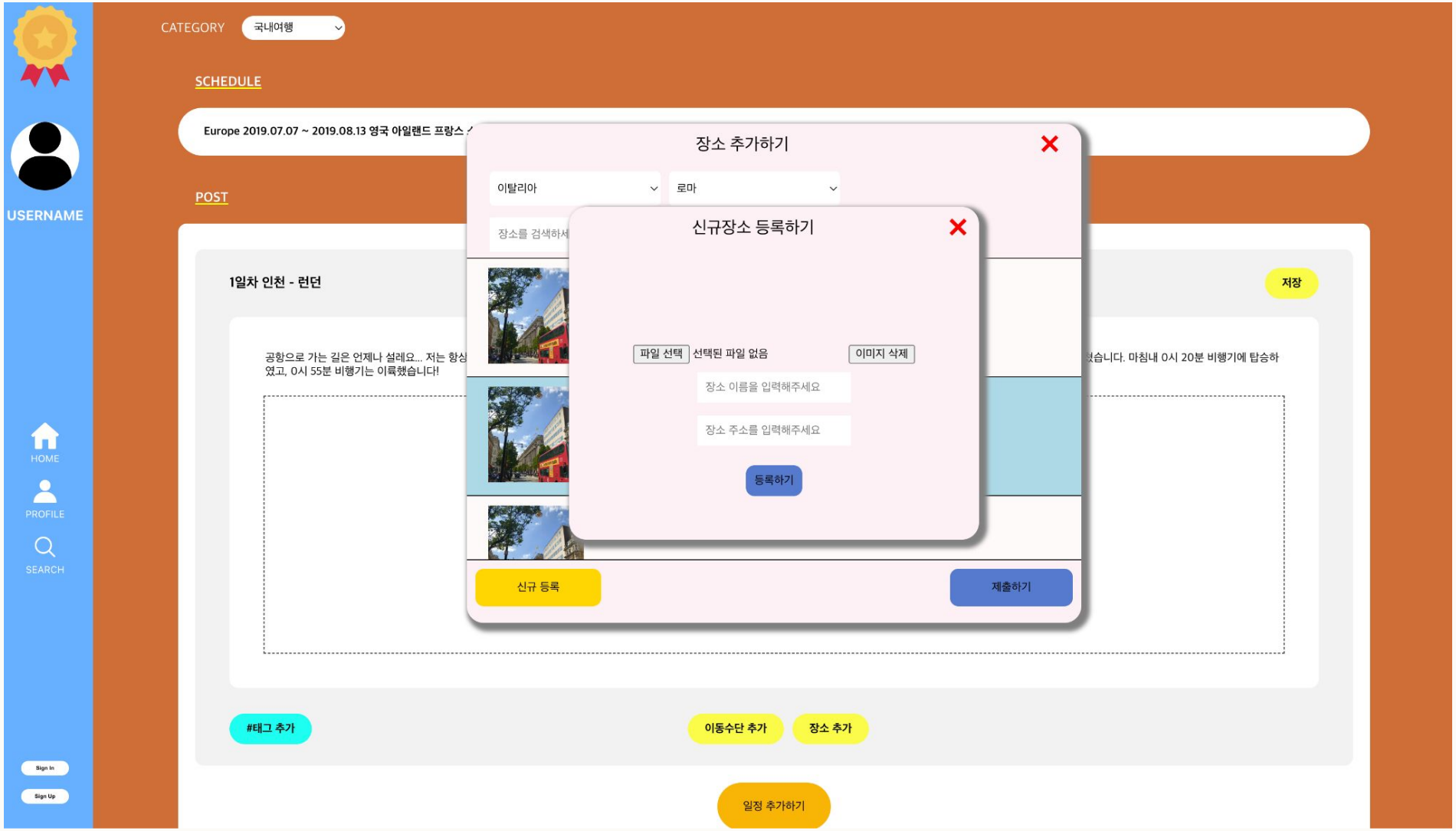
3 페이지

장소 추가 팝업 - 장소 선택



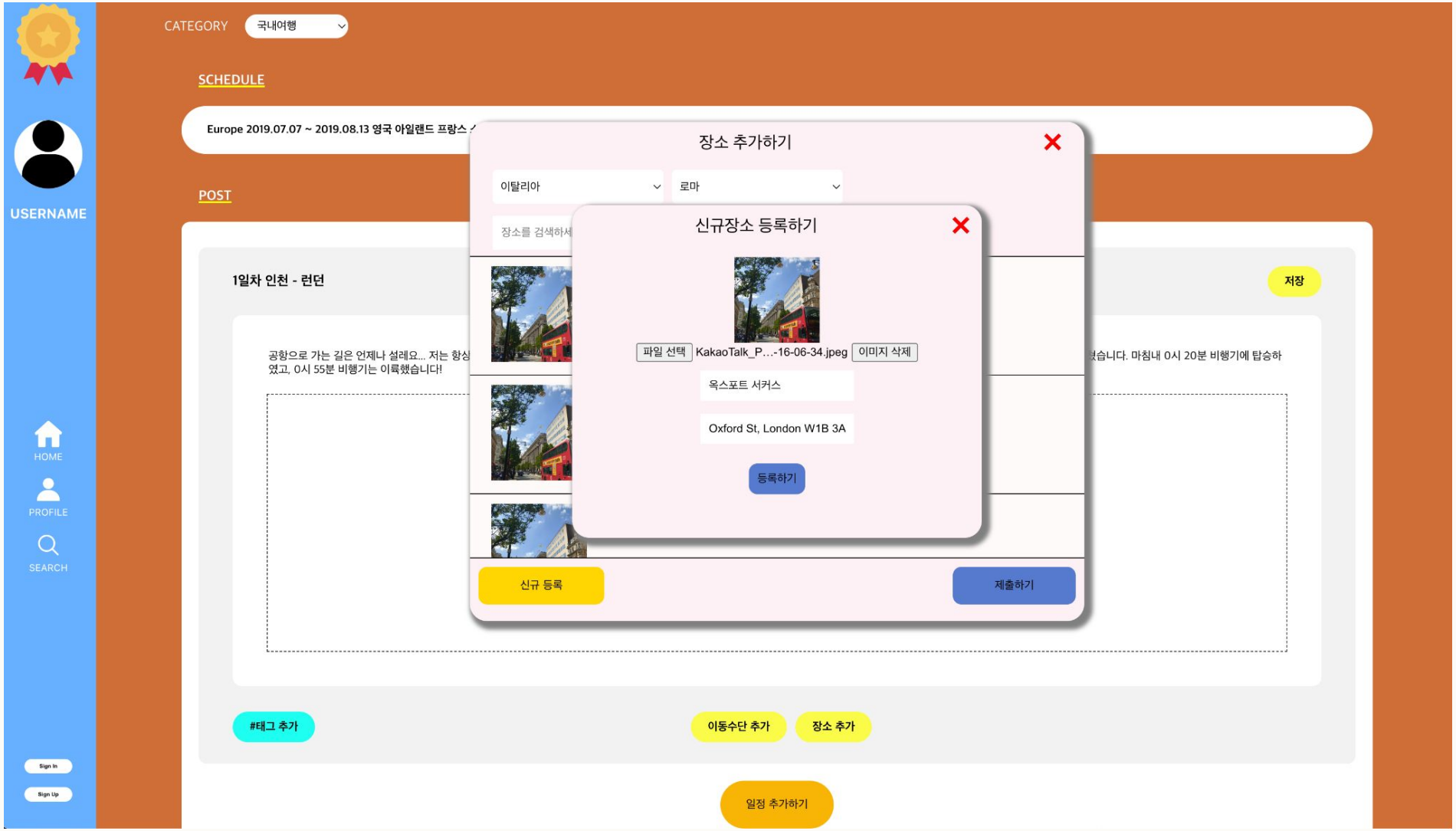
3 페이지 구성도

신규 장소 등록 팝업



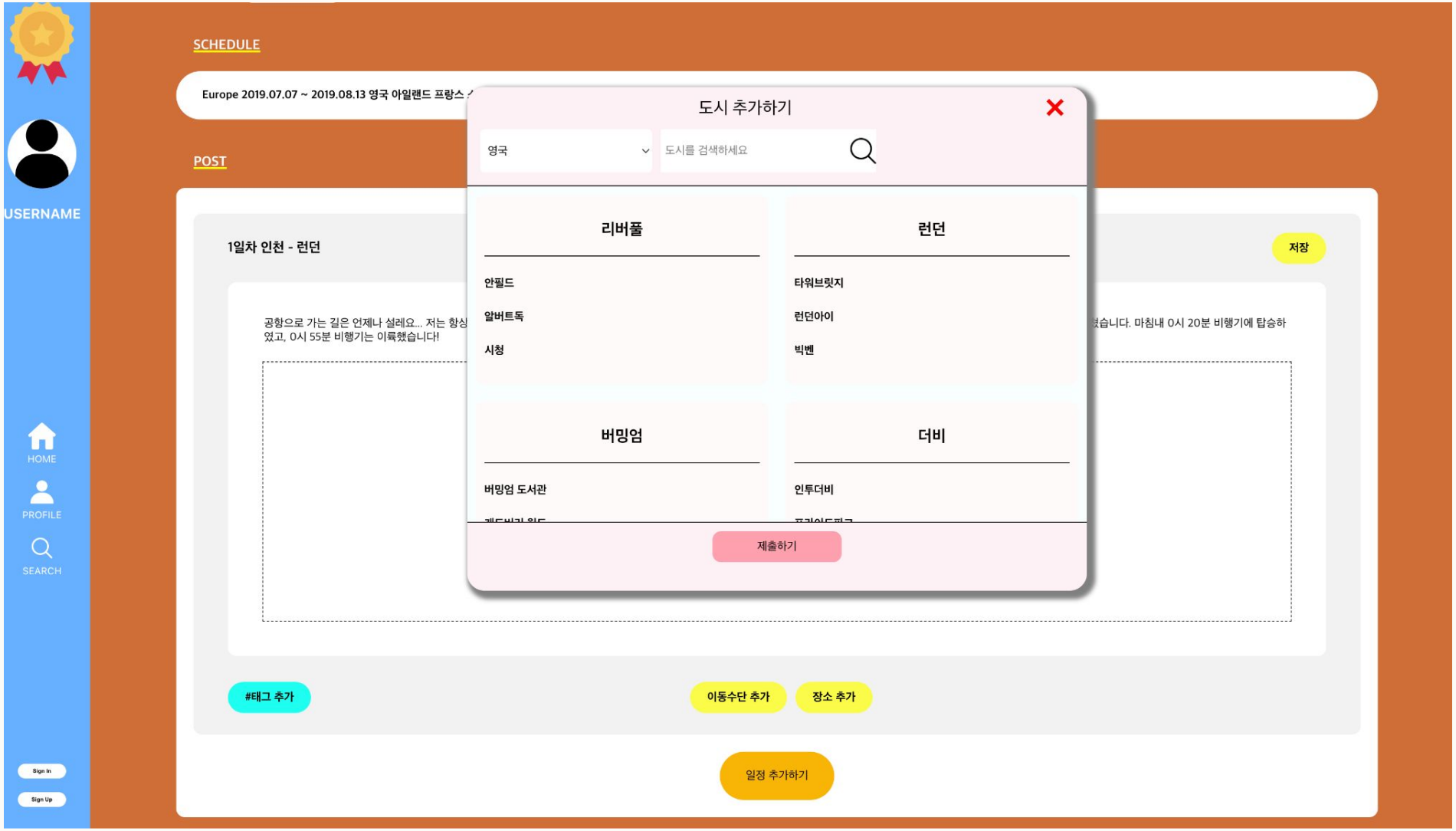
3 페이지 구성도

신규 장소 등록 팝업 - 내용 입력



3 페이지

도시 추가 팝업

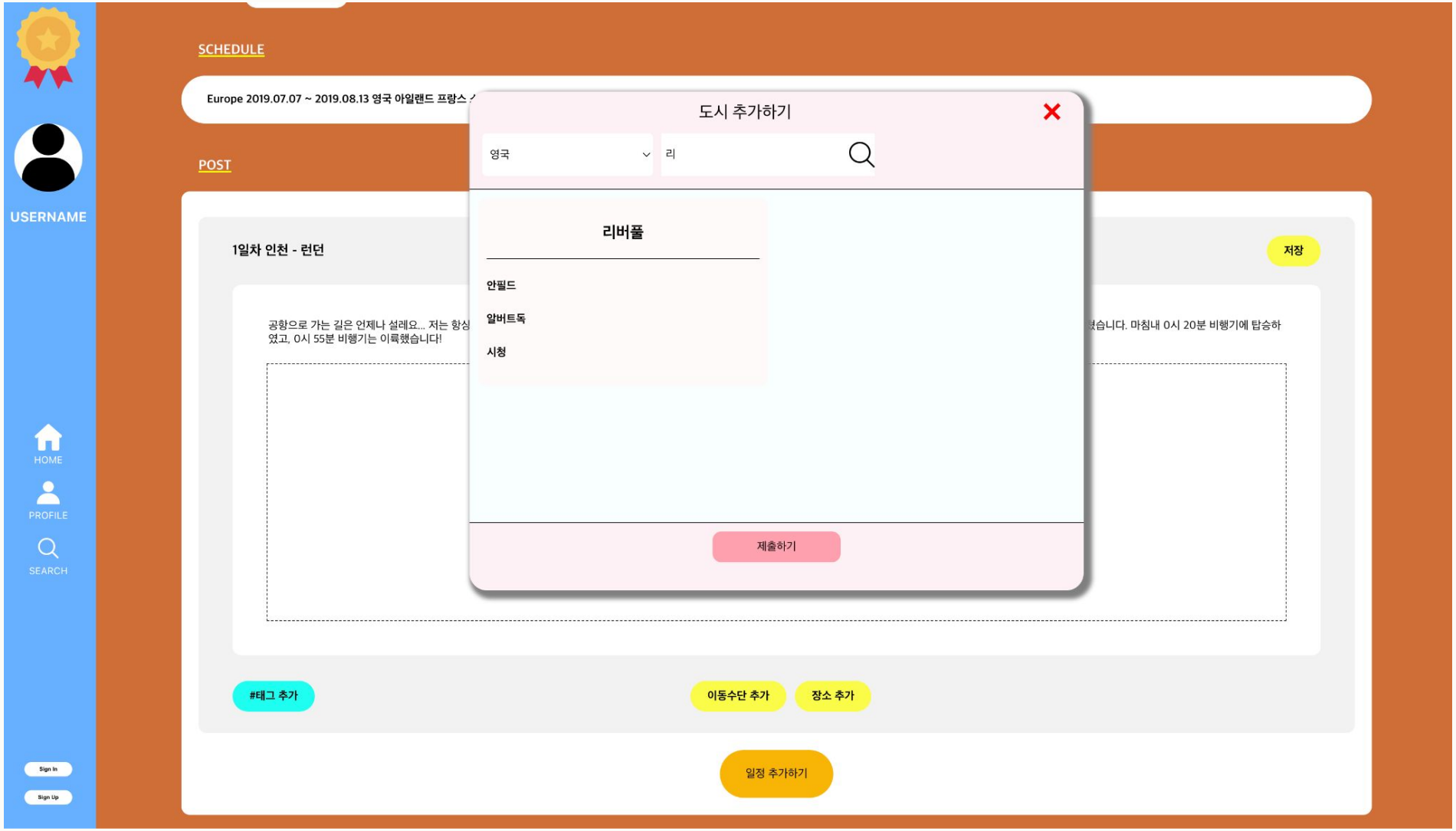

USERNAME
HOME
PROFILE
SEARCH
Sign In
Sign Up
SCHEDULE
Europe 2019.07.07 ~ 2019.08.13 영국 아일랜드 프랑스
POST
1일차 인천 - 런던
저장
공항으로 가는 길은 언제나 설레요... 저는 항상
었고, 0시 55분 비행기는 이륙했습니다!
었습니다. 마침내 0시 20분 비행기에 탑승하
#태그 추가
이동수단 추가
장소 추가
일정 추가하기
도시 추가하기
영국
도시를 검색하세요
리버풀
안필드
알버트독
시청
런던
타워브릿지
런던아이
빅벤
버밍엄
버밍엄 도서관
더비
인투더비
제출하기

3 페이지 구성도

도시 추가 팝업 - 도시 선택

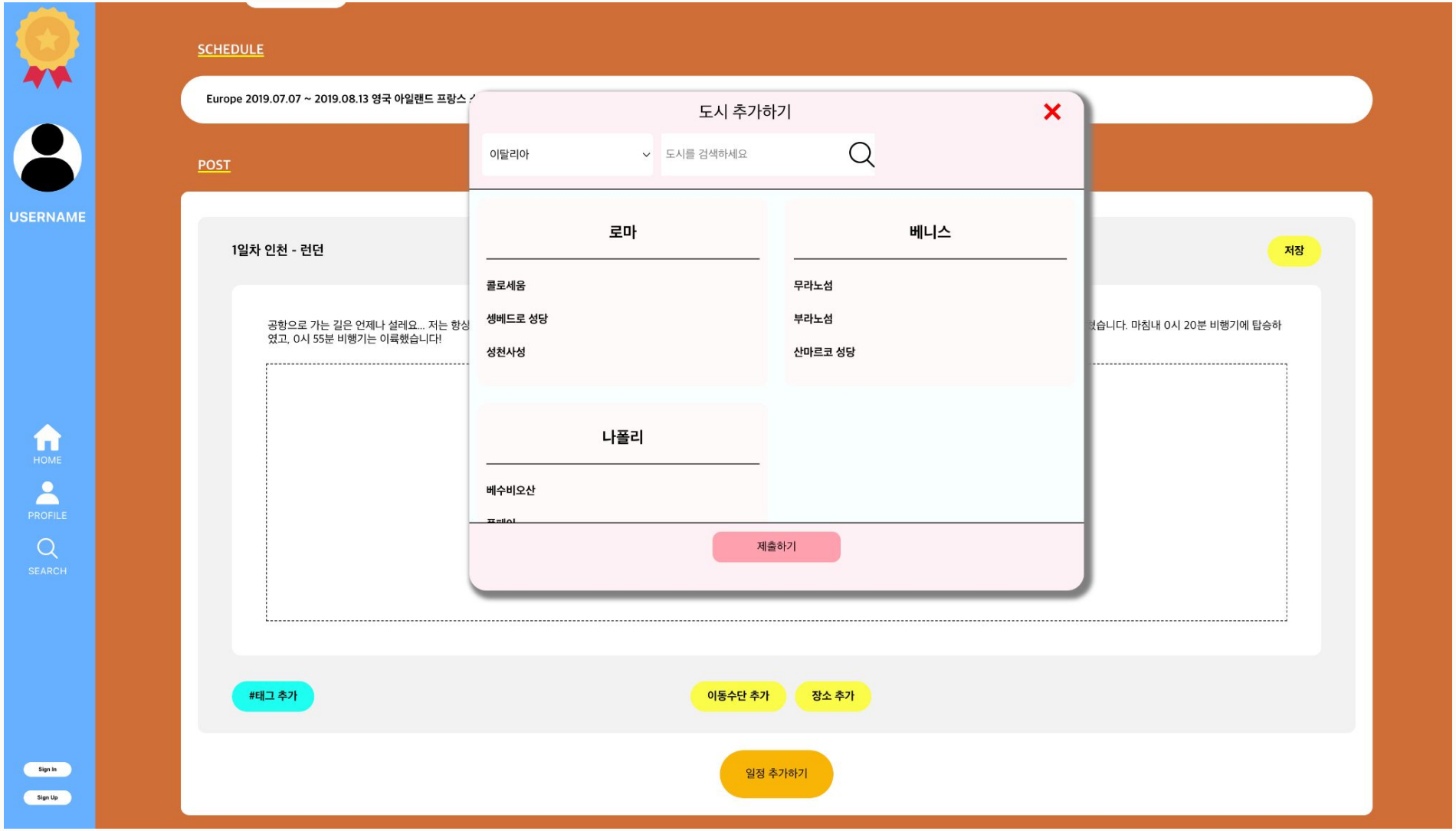
3 페이지

도시 추가 팝업 - 도시 검색



3 페이지 구성도

도시 추가 팝업 - 나라 변경



4

데이터베이스 디자인

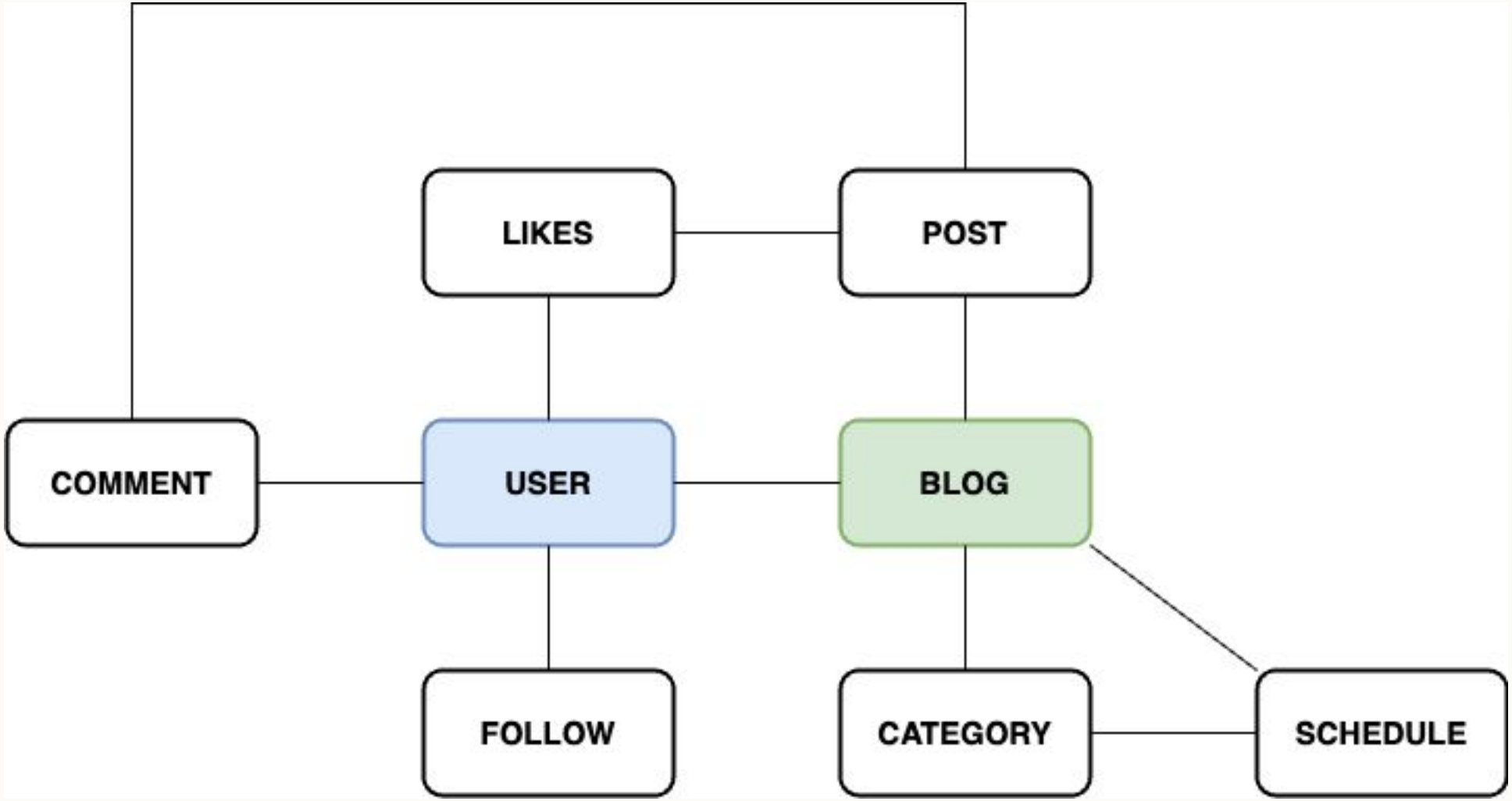
4 데이터베이스 디자인

사용 데이터베이스 - MariaDB

MariaDB

4 데이터베이스 디자인

ERD



mariaDB(RDBMS)

…	createdTime	updatedTime	deleted

Entity를 통해 JPA로 테이블 생성 시, Base Entity 상속받아 생성

4 데이터베이스 디자인

DB 테이블 명세

USER(Entity이름은 SiteUser)

칼럼명	칼럼 설명	Type	Not Null	Default
UserNo	PK	int	O	**AUTO_INCREMENT**
UserId	사용자 아이디	varchar(20)	O	unique
UserPassword	사용자 비밀번호	varchar(200)	O	
UserPhone	사용자 전화번호	varchar(128)	O	unique
UserBirth	사용자 생일	Date	O	
UserEmail	사용자 메일 주소	varchar(20)	O	unique

4 데이터베이스 디자인

DB 테이블 명세

PROFILE

칼럼명	칼럼 설명	Type	Not Null	Default	REFERENCE
ProfileNo	PK(프로파일 넘버)	int	O		**AUTO_INCREMENT**
UserNo	FK(유저 넘버)	int	O		USER(UserNo)
UserName	사용자 이름	varchar(12)	O	default_string	
PhotoUrl	사용자 프로필 사진	varchar(100)	O	default_img	
Introduction	사용자 자기소개	varchar(300)		null	
Tags	사용자 선호 태그	varchar(300)		null	
FollowerNumber	사용자 팔로우 수	int	O	0	
FollowingNumber	사용자 팔로잉 수	int	O	0	
Badge	사용자 뱃지	varchar(300)		null	

4 데이터베이스 디자인

DB 테이블 명세

FOLLOW

칼럼명	칼럼 설명	Type	Not Null	Default	REFERENCE
FollowNo	PK(팔로우 넘버)	int	O		**AUTO_INCREMENT**
FollowerId	FK 팔로워의 이름	varchar(12)	O		PROFILE(UserName)
FollowingId	FK 팔로잉의 이름	varchar(12)	O		PROFILE(UserName)

BLOG

칼럼명	칼럼 설명	Type	Not Null	Default	REFERENCE
BlogNo	PK(블로그 넘버)	int	O		**AUTO_INCREMENT**
Title	블로그 제목	varchar(50)	O		
Subtitle	블로그 부제	varchar(50)		null	
UserNo	FK 블로그 주인의 넘버	int	O		USER(UserNo)

4 데이터베이스 디자인

DB 테이블 명세

POST

칼럼명	칼럼 설명	Type	Not Null	Default	REFERENCE
PostNo	PK(작성글 넘버)	int	O		**AUTO_INCREMENT**
BlogNo	FK 블로그 넘버	int	O		BLOG(BlogNo)
Title	글 제목 50자 제한	varchar	O		
Content	글 내용 200자 제한	TEXT	O		
ThumbnailUrl	게시물의 첫 이미지	varchar	O		
ImgUrl	게시물 내 이미지들	JSON		null	
Url	게시물 URL	varchar	O		
View	조회수	int	O	0	
Date	작성/수집 날짜	date	O	now()	
Transportation	이동수단	varchar	O		
Place	장소	varchar	O		
CategoryNo	FK 카테고리 넘버	int	O		CATEGORY(CategoryNo)
Tags	태그를 저장(태그 이름을 쉼표로 구분)	varchar(300)		null	

4 데이터베이스 디자인

DB 테이블 명세

CATEGORY

칼럼명	칼럼 설명	Type	Not Null	Default	REFERENCE
CategoryNo	PK(카테고리 넘버)	int	O		**AUTO_INCREMENT**
Name	카테고리 이름(대륙)	varchar(12)	O		unique

COMMENT

칼럼명	칼럼 설명	Type	Not Null	Default	REFERENCE
CommentNo	PK(덧글 넘버)	int	O		**AUTO_INCREMENT**
UserNo	FK 작성자 넘버	int	O		USER(UserNo)
UserName	FK 작성자 이름	varchar(12)			PROFILE(UserName)
PostNo	FK 작성글 넘버	int	O		POST(PostNo)
Content	댓글 내용	TEXT	O		
Date	작성 날짜	timestamp	O	now()	

4 데이터베이스 디자인

DB 테이블 명세

SCHEDULE

칼럼명	칼럼 설명	Type	Not Null	Default	REFERENCE
ScheduleNo	PK(일정 넘버)	int	O		**AUTO_INCREMENT**
CategoryNo	FK 카테고리 넘버	int	O		CATEGORY(CategoryNo)
BlogNo	FK 블로그 넘버	int	O		BLOG(BlogNo)
Content	일정 내용	varchar(300)	O		

4 데이터베이스 디자인

DB 테이블 명세

LIKES

칼럼명	칼럼 설명	Type	Not Null	Default
PostId	FK 좋아요가 눌린 게시글	int	O	POST(PostId)
UserId	FK 좋아요를 누른 유저	int	O	USER(UserNo)
PostId,UserId	CK(복합키)		O	

4 데이터베이스 디자인

DB 테이블 명세

AUTH

SOCIALLOGIN

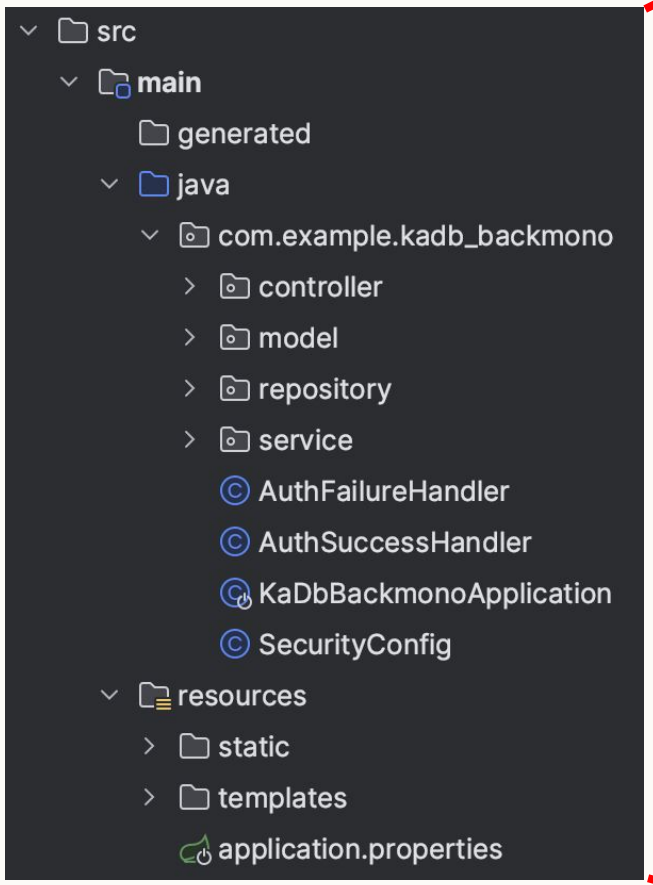
칼럼명	칼럼 설명	Type	Not Null	Default	REFERENCE
SocialLoginNo	PK(소셜로그인 넘버)	int	O		**AUTO_INCREMENT**
UserNo	FK 유저 넘버	int	O		USER(UserNo)
ExternalId	외부 아이디	varchar(64)	O		
AccessToken	액세스 토큰	varchar(256)	O		

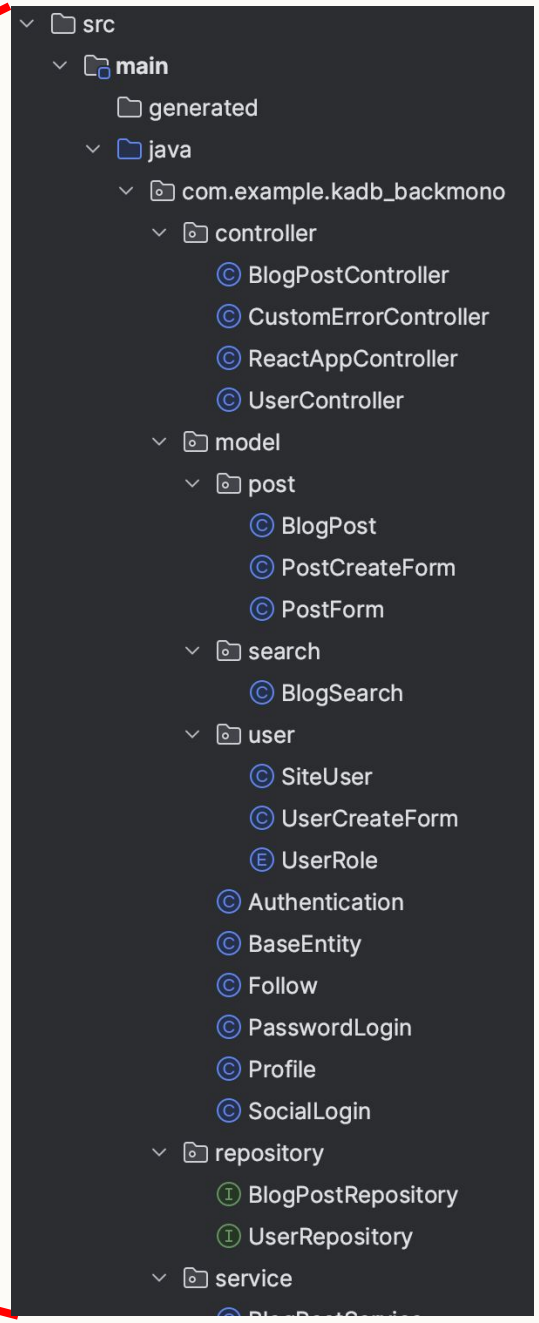
5

BE 구현

5 SpringBoot 파일 구조

MVC 패턴





5 Spring 파일 구조

Controller

```java
import java.util.HashMap;
import java.util.Map;

@RequiredArgsConstructor
@Controller
@RequestMapping("/user")
public class UserController {

    private final UserService userService;

    @PostMapping("/signup")
    public ResponseEntity<Map<String, String>> signup(@Valid @RequestBody User
        Map<String, String> response = new HashMap<>();

        System.out.println("User ID: " + userCreateForm.getUserid());
        System.out.println("Email: " + userCreateForm.getEmail());
        System.out.println("Password: " + userCreateForm.getPassword());
        System.out.println("Phone: " + userCreateForm.getPhone());
        System.out.println("Birthday: " + userCreateForm.getBirthday());

        try {
            userService.create(userCreateForm.getUserid(), userCreateForm.getE
            response.put("status", "success");
            response.put("message", "회원가입이 완료되었습니다.");
            return ResponseEntity.ok(response);
        } catch (DataIntegrityViolationException e) {
            response.put("status", "error");
            response.put("message", "이미 등록된 사용자입니다.");
            return ResponseEntity.badRequest().body(response);
        } catch (Exception e) {
            response.put("status", "error");
            response.put("message", e.getMessage());
            return ResponseEntity.badRequest().body(response);
        }
    }
```

```java
package com.example.kadb_backmono.controller;

import org.springframework.boot.web.servlet.error.ErrorController;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class CustomErrorController implements ErrorController {
    1 usage
    private final String ERROR_PATH = "/error";

    @GetMapping(ERROR_PATH)
    public String redirectRoot(){
        return "index.html";
    }

    no usages
    public String getErrorPath(){
        return null;
    }
}
```

서비스별 컨트롤러 구성 및
리액트가 **SPA(Single Page Application)**를
전제로 하고 있기에, CustomErrorController을 만들어
지정된 경로에 따라 index.html 페이지 리디렉션을 실행.

5 Spring 파일 구조

Model

```java
package com.example.kadb_backmono.model.user;

import com.example.kadb_backmono.model.BaseEntity;
import jakarta.persistence.*;
import lombok.Data;

import java.util.Date;

21 usages
@Data
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn(name = "DTYPE")
@Table(name = "USER")
public class SiteUser extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "UserNo")
    private Integer no;

    @Column(name = "UserId", nullable = false, unique = true, length = 20)
    private String id;

    @Column(name = "UserPassword", nullable = false, length = 200)
    private String password;

    @Column(name = "UserEmail", nullable = false, unique = true, length = 20)
    private String email;

    @Column(name = "UserPhone", nullable = false, unique = true, length = 128)
    private String phone;

    @Column(name = "UserBirthday", nullable = false, length = 128)
    private Date birthday;
}
```

JPA

Java Persistence API

HIBERNATE

5 Spring 파일 구조

Repository & Service

```java
package com.example.kadb_backmono.service;

import com.example.kadb_backmono.model.user.SiteUser;
import com.example.kadb_backmono.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import lombok.RequiredArgsConstructor;

import java.util.Date;

2 usages
@RequiredArgsConstructor
@Service
public class UserService {

    @Autowired
    private final UserRepository userRepository;

    @Autowired
    private final PasswordEncoder passwordEncoder;

    public SiteUser create(String userid, String email, String password, String p
        SiteUser user = new SiteUser();
        user.setId(userid);
        user.setEmail(email);
        user.setPassword(passwordEncoder.encode(password));
        user.setPhone(phone);
        user.setBirthday(birthday);
        this.userRepository.save(user);
        return user;
    }
}
```

```java
package com.example.kadb_backmono.repository;

import ...

4 usages
@Repository
public interface UserRepository extends JpaRepository<SiteUser, Integer> {
    1 usage
    Optional<SiteUser> findById(String userId);
}
```

```java
package com.example.kadb_backmono.repository;

import ...

2 usages
@Repository
public interface BlogPostRepository extends JpaRepository<BlogPost, Integer>{
}
```

JPA 사용을 위한 **레포지토리** 구성 및
DB에서 데이터를 받아와서
비즈니스 로직을 실행하는 **서비스** 구성

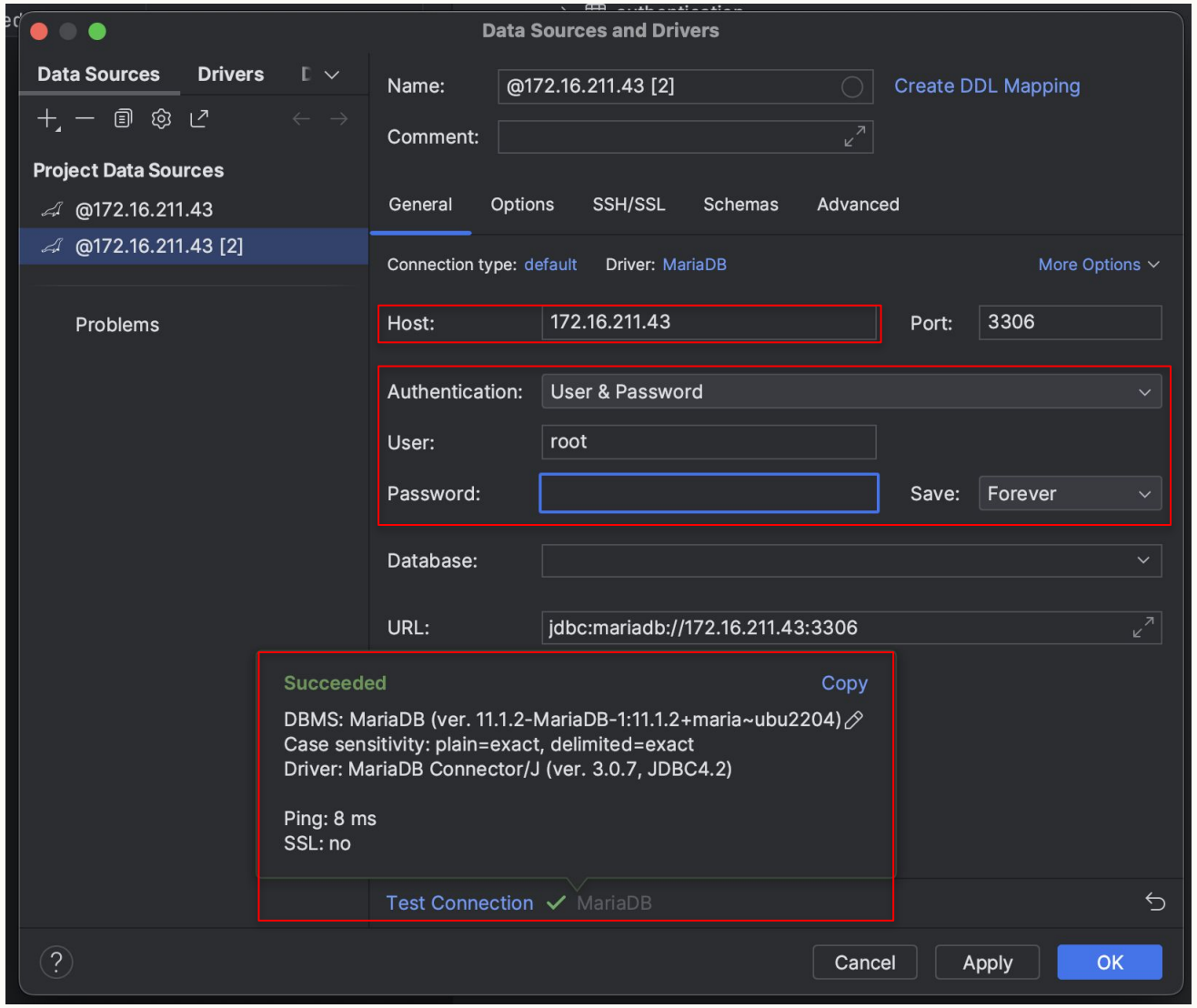
5 Spring 파일 구조

회원 관리 및 인증

```java
//    private final AuthFailureHandler authFailureHandler;

    public SecurityConfig(AuthSuccessHandler authSuccessHandler, AuthFailureHan
        this.authSuccessHandler = authSuccessHandler;
//        this.authFailureHandler = authFailureHandler;
    }

    @Bean
    SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .authorizeHttpRequests((authorizeHttpRequests) -> authorizeHttpReq
                    .requestMatchers(new AntPathRequestMatcher("/**")).permi
            .csrf((csrf) -> csrf
                    .ignoringRequestMatchers(new AntPathRequestMatcher("/**"
            .formLogin((formLogin) -> formLogin
                    .defaultSuccessUrl("/main")
                    .failureUrl( authenticationFailureUrl: "/login?error=true")
                    .loginPage("/login")
                    .loginProcessingUrl("/login")
                    .successHandler(authSuccessHandler)
                    .permitAll())
            .logout((logout) -> logout
                    .logoutRequestMatcher(new AntPathRequestMatcher("/user/l
                    .logoutSuccessUrl("/")
                    .invalidateHttpSession(true))
        ;
        return http.build();
    }

    @Bean
    PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

    @Bean
    AuthenticationManager authenticationManager(AuthenticationConfiguration au
        return authenticationConfiguration.getAuthenticationManager();
```

5 데이터베이스 연동 확인

MariaDB 연동



5 데이터베이스 연동 확인

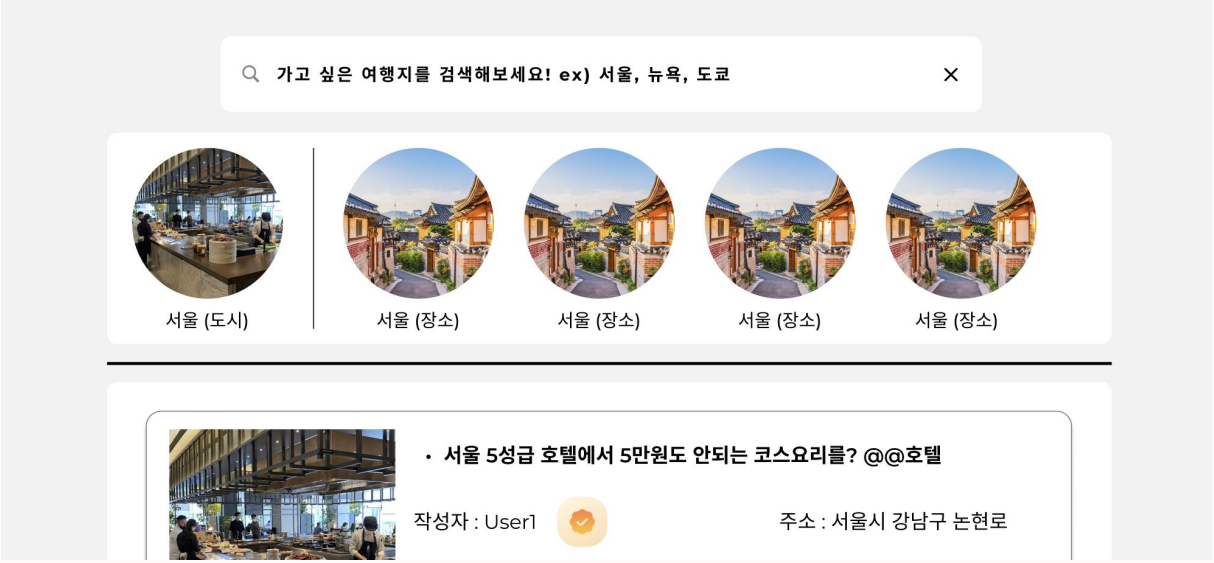
예시 -> 유저 데이터베이스

	pe	user_no	created_time	deleted	updated_time	user_id	user_email	user_password	user_birthday	user_phone
	ser	7	2023-10-30 20:34:23.625…	<null>	2023-10-30 20:34:23.625…	redco7	sungminwer22@naver.con	$2a$10$Mc2PmFOFE.7CrYPJ…	2023-10-03 00:00:00.000…	01022222222
2	ser	8	2023-10-31 10:50:43.488…	<null>	2023-10-31 10:50:43.489…	redco8	test@gmail.com	$2a$10$AcQl1eF/kaLoxW/W…	2023-10-05 00:00:00.000…	01012345678

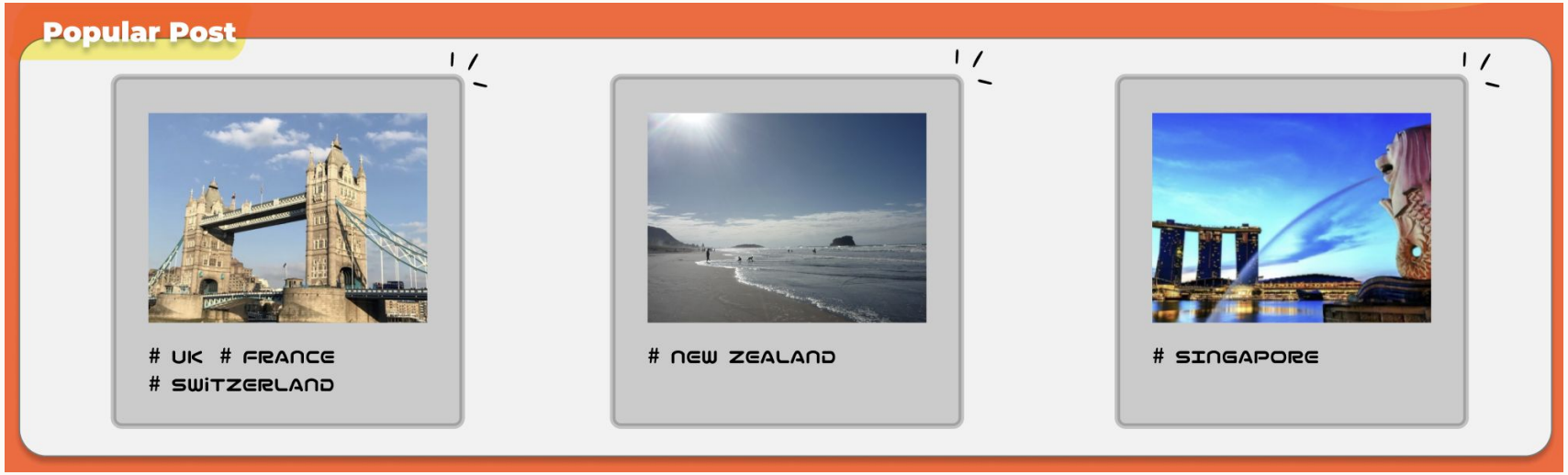
- user
  - columns 10
    - dtype varchar(31)
    - user_no int (auto increment)
    - created_time datetime(6)
    - deleted bit = b'1'
    - updated_time datetime(6)
    - user_id varchar(20)
    - user_email varchar(50)
    - user_password varchar(200)
    - user_birthday date
    - user_phone varchar(128)
  - keys 3
  - indexes 3

5 추천시스템 및 Ai 기술 활용

FastAPI 개발



FastAPI



Post 작성시 추가하는
Tag기반으로 관련 글, 장소
추천시스템을 구현.

썸네일 자동 생성 ai를 활용한 글
내용에 따른 썸네일 생성을 구현

6

Trouble Shooting

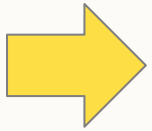
6 Trouble Shooting

Front-End

- react에서 classname이 같은 css item은 다른 css 파일에 있더라도 덮어 씌어진다.

css파일을 따로 만들지 말고 js 파일 안에서 styled-component를 import하여 스타일 적용

- 이미지 파일 등을 public 폴더에서 가져오면 경로 설정이 힘들다.



각 이미지 파일은 js 파일이 포함된 폴더에 적재하여 사용

- 로그인을 담당하는 Spring Securitiy으로 데이터 post 요청할 때 json 타입의 데이터는 불가하다.

axios.post의 content-type을 “application/x-www-form-urlencoded”으로 설정하고, 전송할 데이터를 객체로 정의 후 qs.stringfy(객체)로 하여 서버로 데이터를 post 요청한다.

6 Trouble Shooting

Back-End

- 데이터베이스 연결 문제(백엔드 서비스와 데이터베이스에 연결할 수 없음)

데이터베이스 서버 상태 확인, 데이터베이스 url 확인

- API 문제(API 엔드 포인트에서 예상하지 못한 오류, 데이터 형식 문제)

로그 확인 및 오류 핸들링 개선, API 요청 확인(프론트와 백간의 소통),
응답 디버깅

- 아키텍처 고려 문제(Database 사용 용도에 따른 결정 문제)

용도별 DB 사용 (RDBMS, NoSQL), 카프카를 통한 이중화된 데이터베이스 연동 고려중

7

팀원 소개

7 팀원 소개

윤건우
컴퓨터공학과
201833937
PM, 자료조사
BE

강석훈
컴퓨터공학과
201835605
PPT
FE

김성민
소프트웨어학과
201735815
발표
BE

장현화
컴퓨터공학과
201935334
자료조사
FE

채호기
소프트웨어학과
201835535
PPT
FE

ⓒSaebyeol Yu. Saebyeol's PowerPoint

Thanks !