설계 분석서

“씬 스토리 맵_ 원하는 장소의 기억을 기록하고 열람하다.”

2025년 05월 25일

씬 스토리텔러 팀

(정동진 나승엽 오민규 정재운 김성인 고차통)

전남대학교

문서 정보

구분	소속	성명	날짜	서명
작성자	씬 스토리텔러	정동진	2025.05.25	
검토자	씬 스토리텔러	오민규	2025.05.25	
	씬 스토리텔러	정재운	2025.05.25	
	씬 스토리텔러	김성인	2025.05.25	
	씬 스토리텔러	고차통	2025.05.25	
	씬 스토리텔러	나승엽	2025.05.25	

개정 이력

개정 일자	버전	개정 내역	작성자	승인자
			검토자	관리자
2025.05.25.	1.0	초안 작성	정동진	김진술
			오민규	김진술
2025.05.25.	1.1	블록 다이어그램 및 UML 순서 다이어그램	정동진	김진술
			오민규	김진술
2025.05.25.	1.2	UML 클래스 다이어그램	정동진	김진술
			오민규	김진술
2025.05.25.	1.3	문서 정렬 및 점검	정동진	김진술
			오민규	김진술
2025.05.25.	1.4	오류 확인 및 최종 점검	정동진	김진술
			오민규	김진술

목 차

1. 개요

1.1. 목표

- ▶ 본 문서는 “씬 스토리 맵”의 요구사항을 검토하고 정의하는 것을 목적으로 한다.
- ▶ “씬 스토리 맵”이란, 여행을 하고 싶어하는 모든 사용자들의 경험하고 싶은 경험들에 기반한 적절한 장소 탐색에 도움을 주고자 하는 웹 사이트 개발로써, 세부 목표는 해당 웹 사이트를 통하여 해당 웹 사이트를 이용하는 모든 사용자들이 자신이 가 보고 싶은 장소를 다른 사용자들의 경험들을 기록해 둔 피드를 통해 알아내거나, 자신이 경험한 것을 피드를 통해 기록하고 지도를 살펴 보며 자유롭게 기록할 수 있게 하는 기능을 제공하는 것이다. 가장 많이 사용하는 기기인 스마트 폰, 노트북, 데스크톱 등 임의의 인터넷이 허용되는 모든 전자기기를 통해 해당 웹 사이트에 접근하고 자신이 원하는 아이디와 패스워드로 로그인과 회원가입을 하여 사용자들이 자신이 경험 위주로 하나 하나 탐색하길 원하는 모든 지도 상의 위치에 대한 다른 사용자들, 혹은 자신이 작성한 피드 목록 정보를 마커를 통해 지도 상에 표기해 제공하고, 직접 원하는 위치에 피드 마커와 함께 자신이 찍었던 경험들이 담긴 사진과 문구를 통해 어떤 경험을 했는지에 대해 원하는 만큼 긴 문장을 작성할 수 있다. 최종 목표로는 서버와 데이터 베이스 구축을 적용하여 모든 웹 서버 사용자들이 방문했을 때 자신이 원하는 시각에 원하는 지도 상의 위치에 즉시 피드를 생성할 수 있게, 또 다른 사용자들이 생성한 피드 목록을 실시간으로 즉시 확인할 수 있게 하는 웹 사이트를 구현하는 것이다.

1.2. 데이터 서버의 범위.

(1) 사용자 요청 처리: 로그인, 회원가입, 피드 작성, 피드 조회, 마커 생성, 프로필 사진과 배경화면 수정, 프로필 문구 수정, 계정 삭제 요청 등 모든 사용자의 상호작용 요청에 대해 flask 서버는 api 형식으로 요청을 수신하고, 해당 요청에 대응되는 각 api, 즉 flask 라우트 함수를 호출해 사용자의 인증 권한 확인 요청, 사용자의 로그인 가능 여부 확인 요청, 사용자의 피드 마커 생성과 대응되는 피드 작성 권한 요청을 수행해 준다.

(2) 데이터 전달 및 변화 범위: flask 서버는 사용자 요청에 따라 필요한 데이터들을 데이터 베이스로부터 조회하거나, 클라이언트로부터 받은 데이터들을 적절히 가공 및 서버의 저장공간 내의 적절한 고유 피드 사진 폴더 경로에 저장한 후 저장된 경로 값을 db 에 송신함으로써 피드 사진의 jpg 형태의 데이터 형태에서 varchar(500) 으로써의 데이터 변환 범위를 갖는다.

(3) 파일 관리의 범위: 사용자가 업로드하는 피드 이미지 등 서버 내의 특정 디렉토리에 저장되며, flask 서버는 이 특정 폴더 내의 상대경로 값을 db 에 송신해 기록하는 요청을 쿼리문으로써 전송하고, 필요 시에 해당 경로로부터 사진 파일을 반환하라는 요청 또한 실행한다.

(4) 보안 및 접근 제어의 범위 설정: 사용자가 입력한 로그인 폼 내 아이디와 비밀번호 데이터 값을 통한 고유 uuid 값으로써의 인증 토큰의 검증, 해당 검증이 통과되는 로그인 입력 폼에 대해서만 로그인 후의 세션을 유지하고, 고유 사용자로써의 접근 권한의 확인이 된 경우에만 자신이 생성한 피드의 수정을 할 수 있게 하거나 프로필을 수정할 수 있게 하는 형식으로써 전체 시스템의 보안성을 유지한다.

(5) api 제공 범위: 프론트 엔드에서 사용이 가능한 restful api 를 제공하며, 각 기능별 api 엔드 포인트를 정의하고 응답 포맷을 통일한다. '응답 코드.메시지.json' 형식의 데이터로 모든 요청의 반환 값을 표준화한 후, 각 라우트 함수가 출력하고자 하는 반환 값들을 해당 형식대로 반환한다.

1.3 참고자료

(1) 21st.dev – Discover, share, and craft UI components

(2) Home – Figma

(3) JavaScript | MDN

(4) Discord | 친구

(5) kinganimal1210/Basic_GitHub: Practice

(6) 2.1 일반 회원가입 및 로그인을 통해 프로필 수정하고, 구글 맵에서 피드용 마커 생성하기.

(7) Welcome to Flask — Flask Documentation (3.1.x)

(8) Cascading Style Sheets

(9) TSX 파일 - React TypeScript 파일 - .tsx 파일이란 무엇이며 어떻게 여나요?

(10) ChatGPT

(11) [Git / Github] 깃허브 클론 (clone) 하는 방법 (깃허브 리포지토리 로컬로 복사)

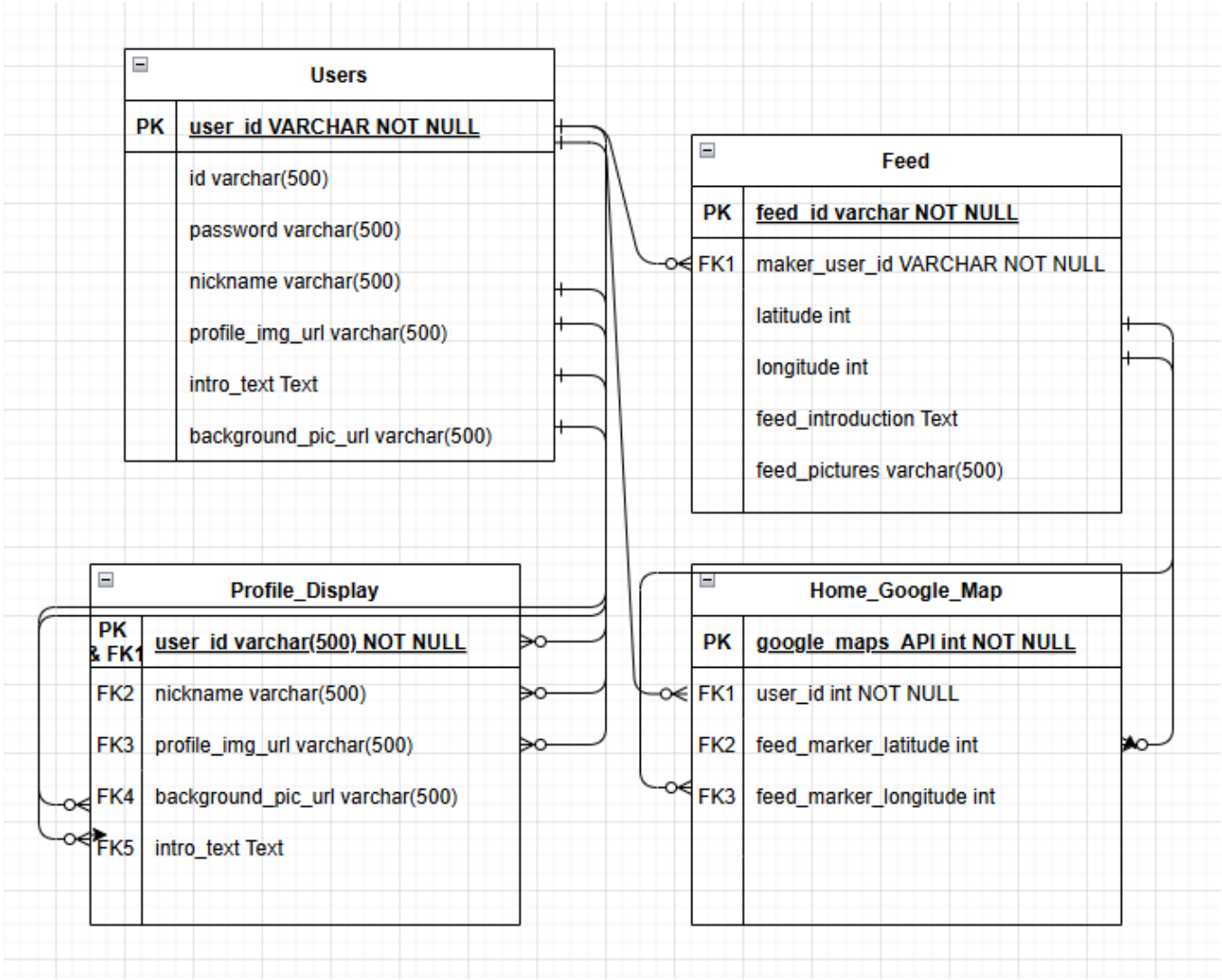
1.4. 용어 및 약어

- (1) API - Application Programming Interface
- (2) TCP - Transmission Control Protocol
- (3) UI - User Interface

2. 시스템 구조 – 블록 다이어그램

2.1. 사용 사례 다이어그램

씬 스토리 맵의 사용 사례에서의 데이터 베이스 전송 시스템은(그림 1)과 같은 구조로 되어 있다.



< 그림 1 : UI 데이터 베이스의 관계 다이어그램. >

데이터 베이스로는 feeds 데이터 베이스, userdb 데이터 베이스가 있다. 먼저 사용자가 Home 화면에 진입하면, feeds 데이터 베이스와 연동된 Home 의 UI 파일에서 feeds 데이터 베이스에 모든 feeds 인스턴스의 latitude, longitude 속성 값을 요청한다. 해당 위도와

경도 위치에 Home UI 파일은 빨강색 마커를 생성하며, 사용자는 로그인하거나 회원가입을 하지 않았더라도 해당 마커를 클릭해서 작성된 피드의 오버레이 창을 띄울 수 있다. 클릭 시 띄워진 오버레이 창에는 다른 사용자들이 자신의 계정으로 로그인한 상태에서 작성한 피드들의 사진 목록들이 등장해야 한다. 따라서 만들어진 피드를 업로드하는 UI 를 담당하는 feed_display UI 파일은 feeds 데이터 베이스에 해당 위도와 경도 위치에 생성된 모든 피드 인스턴스의 feed_introduction: 문구 값과 feed_pictures: 사진 목록이 저장된 파일의 경로 값을 요청한다. 데이터 베이스가 요청된 정보를 보내주면 소개문구는 그대로 각 피드 창마다 출력해 주고, 사진 목록이 저장된 파일의 경로 값에 위치한 서버 내의 위치에서 해당 폴더의 사진들을 순서대로 받아오는 역할을 flask_server.py 파일이 실행한다. 그 결과를 사용자는 해당 위치에 표시된 마커에 대응되는 모든 피드들의 목록을 열람할 수 있게 됨으로써 취하게 된다. 이때 flask_server.py 파일은 별도로 각 피드에 대한 사용자의 maker_user_id 값도 요청한다. 이 값을 users 테이블에 조회하여 해당 테이블로부터 profile_img_url 값과 nickname 값을 받아온 해당 서버는 해당 데이터를 사용자가 피드를 열람하는 feed_display UI 파일에게 전송하고, feed_display UI 파일은 열람하는 피드가 어떤 사용자가 만든 것인지를 표시해 준다. 사용자가 지도 상에서 피드를 생성하고 싶다면 원하는 지도 상의 위치에 마우스 왼쪽 버튼을 클릭해 해당 위치에 마커를 생성하고 동시에 해당 마커에 대응되는 피드 작성 창이 오버레이 창으로써 등장한다. 이를 실행하는 UI 파일은 made_feed UI 파일이다. 이 파일은 즉시 flask_server.py 의 add_feed() 함수를 호출한다. 호출된 add_feed() 함수는 feed 데이터 베이스와 연동되며 사용자가 생성한 피드에 대응되는 고유한 파일명을 가진 사진 목록 저장 폴더를 생성한다. 해당 폴더에는 사용자가 실시간으로 업로드한 피드 사진들이 순서대로 저장되며, 해당 폴더는 서버에 위치한다. 데이터 베이스에는 해당 폴더의 서버 상의 저장소 내 경로명만 저장된다. 비슷한 방식으로 사용자가 작성한 피드의 소개문구는 flask 서버를 통해 데이터 베이스에 Text 데이터 형식으로 저장된다.

2.2. 이 시스템 구조의 장점

- (1) 사용자가 지도를 직접 클릭하여 마커를 생성하고, 그 마커를 기반으로 피드를 생성 및 열람하는 방식은 손쉬운 사용성과 직관적인 사용자 경험(UX)을 제공한다.
- (2) Home, feed_display, made_feed 등 UI가 분리되어 있고, flask_server.py가 데이터 처리 중심 역할을 하며, 각 UI 별로 필요로 하는 데이터의 요청 로직이 명확히 분리돼 유지보수가 쉽다. 즉, 모듈화가 잘 된 구조라 볼 수 있다.
- (3) 로그인을 하지 않아도 피드 열람이 가능하다는 것으로 초기 사용자 진입 장벽을 낮춰 유입률 증가에 기여할 수 있다.
- (4) 마커마다 해당 피드와 사용자 정보를 조인하여 출력하는 방식은 관계형 데이터 모델의 강점을 효과적으로 살린 방법이다.

2.3. 이 시스템 구조의 단점

- (1) flask_server.py가 모든 DB I/O와 파일 처리를 담당하면, 트래픽 증가 시 서버 병목현상이 발생할 수 있는 단일 서버 구조의 한계
- (2) 사용자 조회를 위해 users 테이블과의 조인이 반복적으로 일어나므로, 피드가 많을수록 조인 쿼리 과부하의 가능성으로 성능 저하가 일어날 수 있다.
- (3) 비로그인 상태에서 서버와 통신하며 피드를 불특정 다수가 열람할 수 있다는 점은 보안상 취약점으로 데이터 접근 제어 부족 문제를 유발할 수 있다.

2.4. 구성도 설명

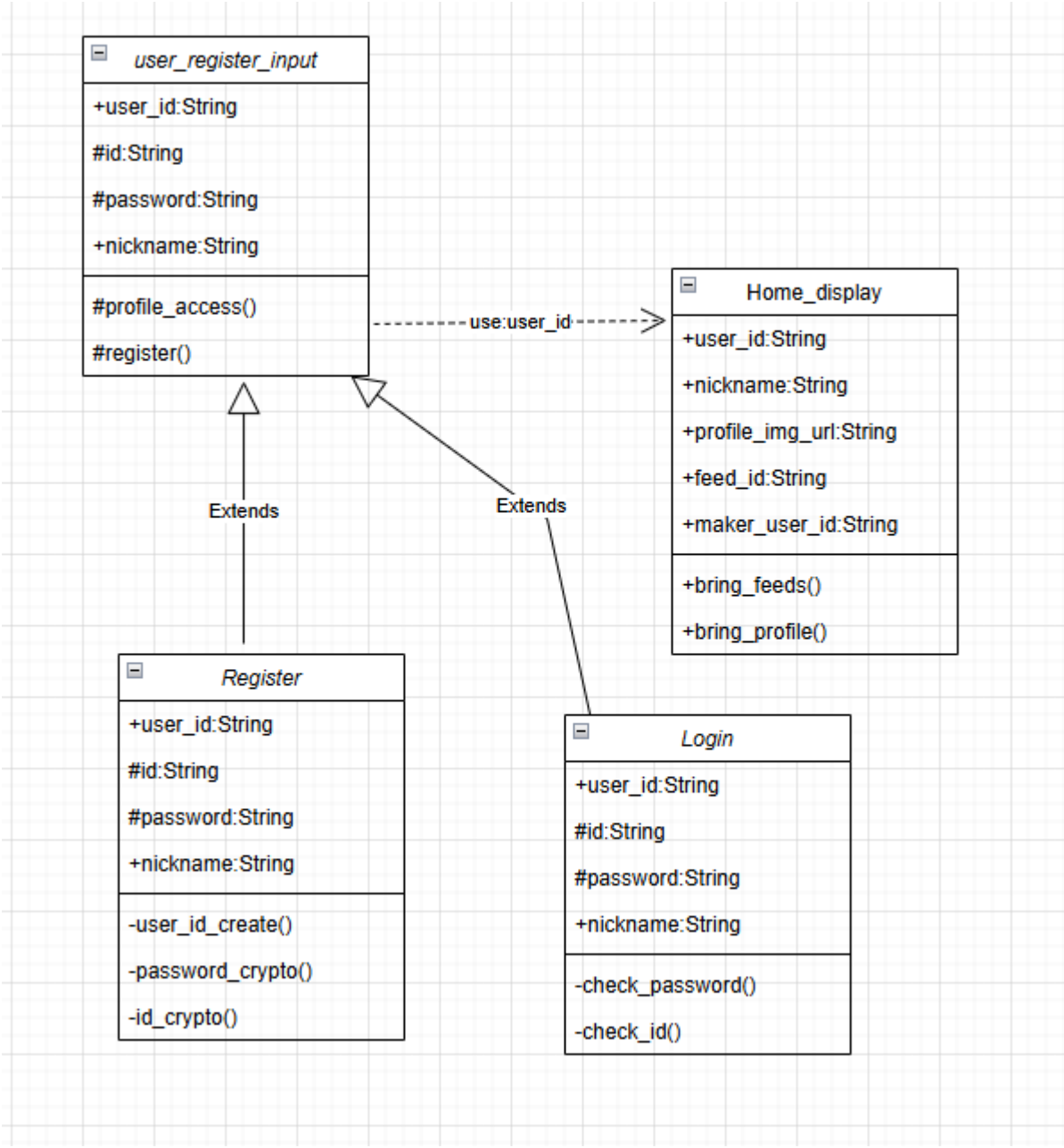
본 웹 애플리케이션은 React.js로 구현된 위치 기반 피드공유 서비스입니다. 사용자는 지도에서 마커를 추가하고 해당 위치에 대한 피드를 등록할 수 있습니다. 구성은 다음과 같습니다.

- **상단 메뉴**: 로고와 사용자 메뉴(Profile, Settings, Logout)로 구성되어 있으며, 사용자의 설정 및 프로필 계정 접근 기능을 제공합니다.

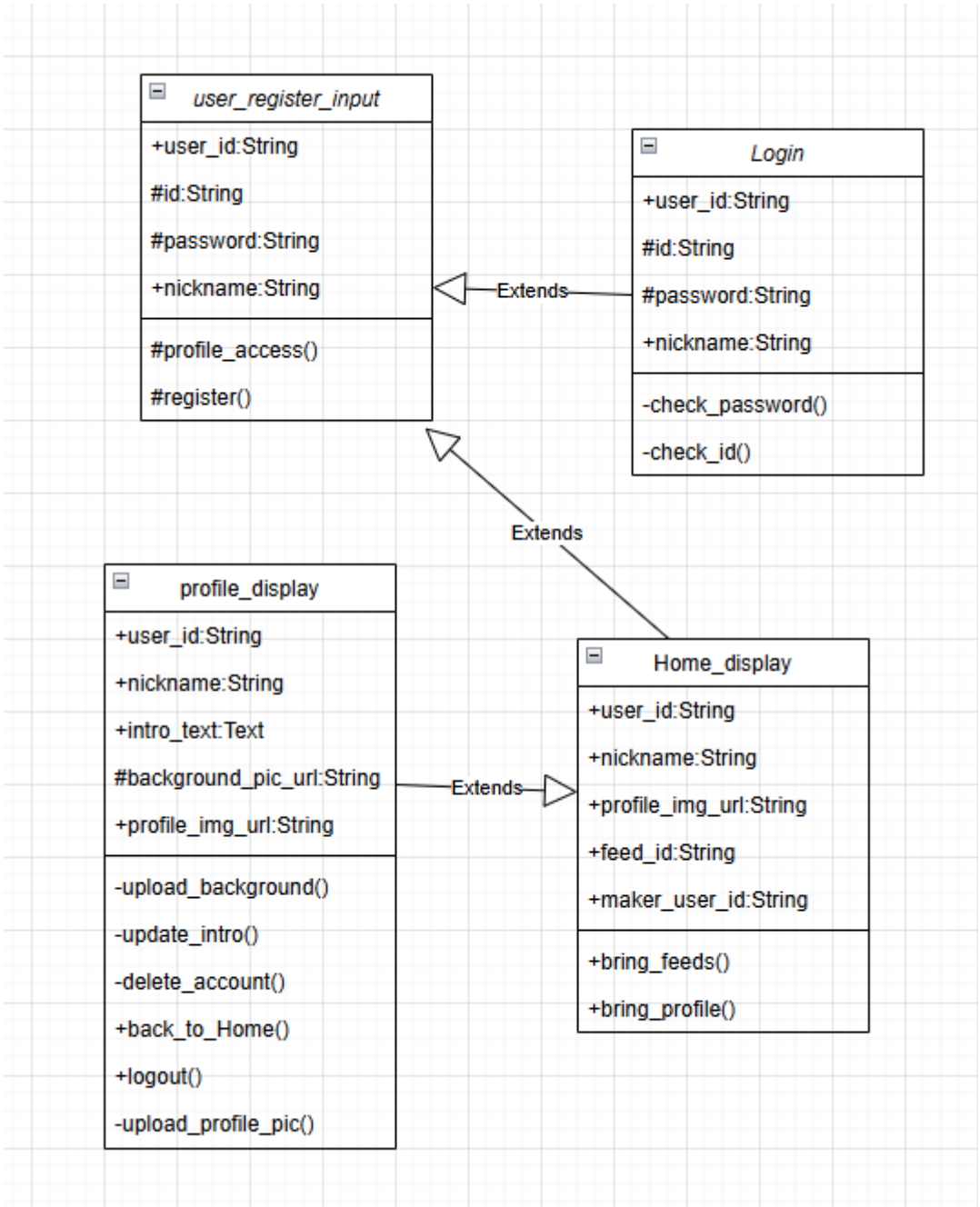
- **중앙 지도 영역**: Google Maps API를 활용해 구현된 지도입니다. 사용자의 피드를 생성함에 따라 마커가 실시간으로 새롭게 표시되며, 원하는 위치에서의 마커 탐색과 마커의 클릭을 통한 연동된 피드 데이터의 조회가 가능합니다.
- **하단 버튼**: "Make Marker!" 버튼을 통해 사용자가 직접 원하는 위치에 마커를 추가할 수 있습니다.
- **서버 통신 구조**: 초기 피드는 Flask 서버에서 호출해 오며, 새로운 피드의 사진 및 소개문구 데이터는 Socket.IO를 통해 실시간으로 수신되고 변경사항이 반영됩니다.

- ** 전체페이지** : 는 HTML 기반의 웹 인터페이스로 구성되며, 지도 중심의 직관적인 사용자 경험을 제공합니다.

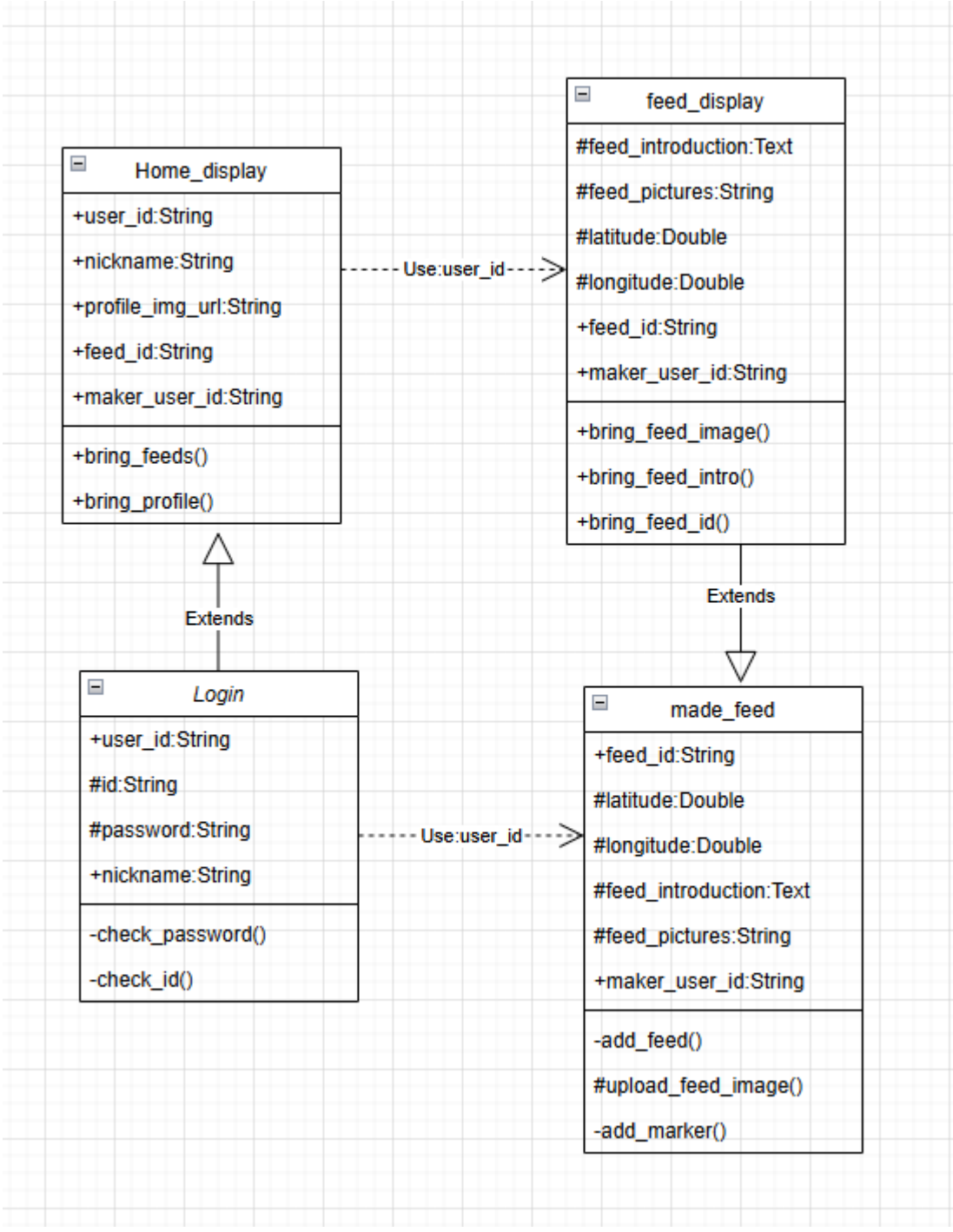
3. 서버 구조



< 회원가입 및 로그인 기능 후 Home 화면으로 이동 >



< 사용자 input 을 통해 로그인 후 홈 화면, 프로필 화면 불러오기 기능 >



< 사용자의 피드 생성 창 작성 후 Home 창에 생성된 피드 마커 업로드하기 >

3-2 데이터 서버의 목적

이 프로젝트의 데이터 서버는 사용자 <--> DB 사이의 중계 역할을 수행하는 웹 백엔드 시스템으로 사용된다.
클라이언트의 다양한 요청(피드 생성, 피드 목록 조회, 이미지 업로드 등)을 처리하기 위해 데이터 베이스 연동을 위해 데이터 서버를 이용한다.

3-3 데이터 서버의 역할

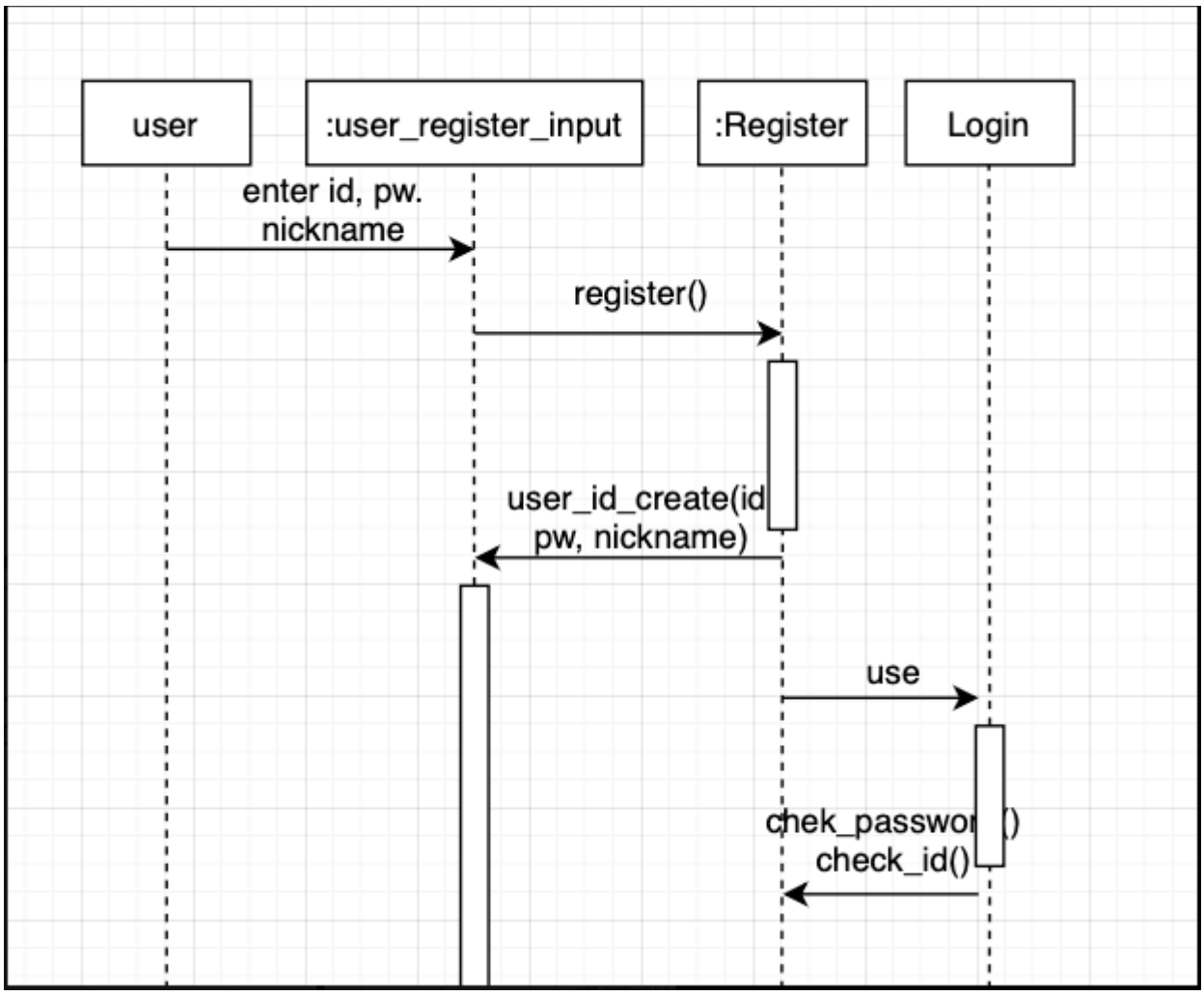
(1) 피드 저장 요청 : 클라이언트로부터 받은 피드 생성 요청을 처리하여, 위도, 경도, 피드의 설명 문구 텍스트, 이미지 파일들을 수신하고 고유 폴더 내에 정리한 뒤 DB에 저장 요청을 보내기 위해 python flask 의 DATA SERVER 를 실행해 서버 연동한다.

(2) 이미지 파일의 url 저장 : 사용자가 업로드한 이미지 파일의 경로를 데이터화 시켜 DATA SERVER에 해당 경로의 값을 인스턴스 값으로 저장하는 쿼리문을 요청하고, 이를 통해 DB의 이미지 url 값의 속성 인스턴스로 업로드한 파일이 저장되는 로컬 서버의 폴더 내 사진의 경로명이 저장된다. 로컬 서버의 피드 id 별 고유 폴더 내에 업로드한 사진이 저장된다.

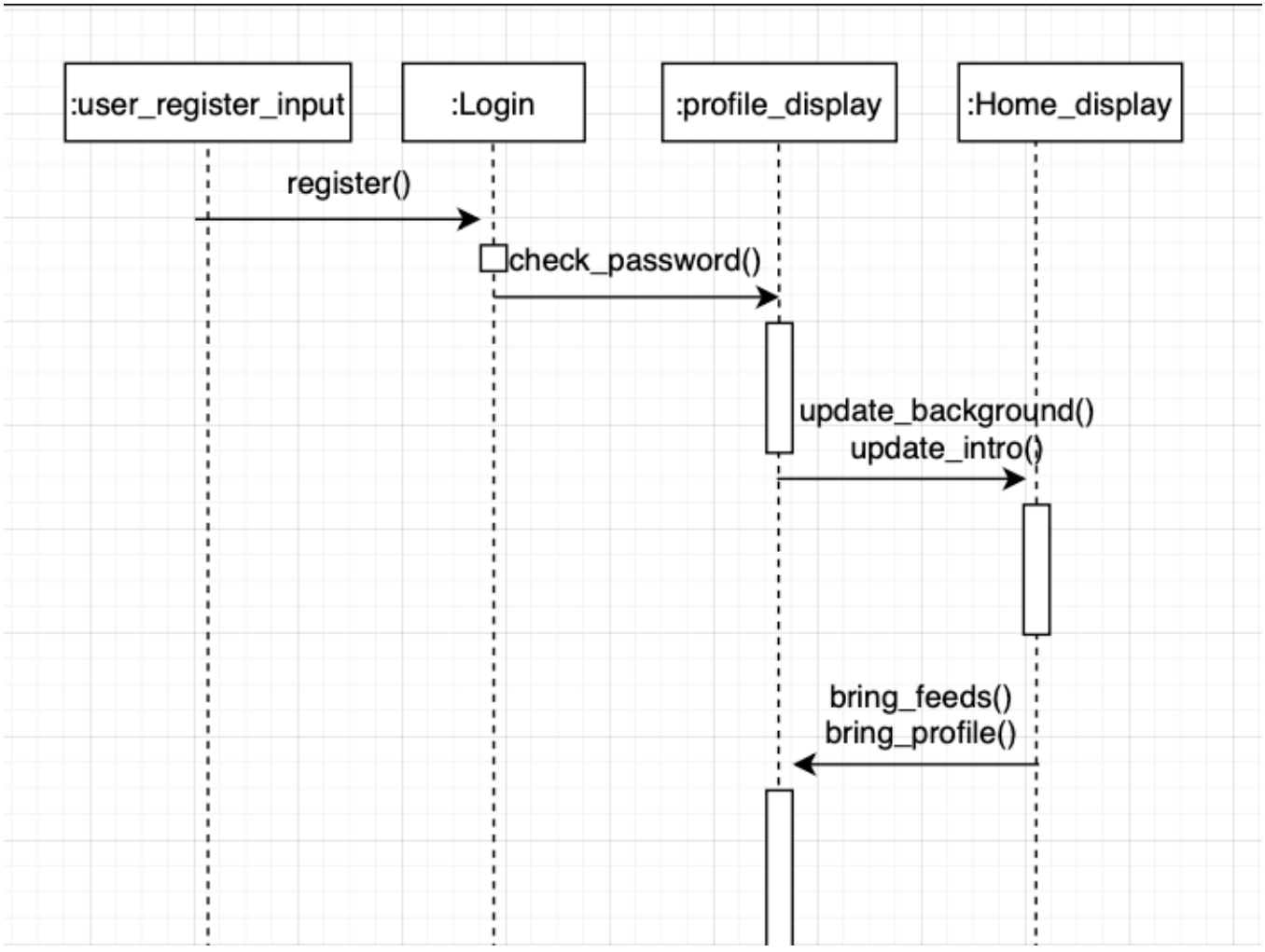
(3) 필요한 정보 요청 : 웹 페이지를 통해 피드 목록을 요청하면 DATA SERVER 를 통해 mysql Database 로 이동되고, DB에서 필요한 정보를 다시 flask SERVER를 통해 웹 페이지 내부로 호출해 온다.
(4) 로그인 / 프로필 사진 호출 : DB의 USER_ID 태그에 USER 의 프로필 사진, 배경화면 사진, 닉네임, 프로필 소개문구 등의 각종 정보가 저장 되어있는데, 이를 불러오기 위해서 flask SERVER 는 쿼리문 작성을 통해 mysql database 에 자신이 필요한 사용자의 정보 데이터를 요청한다.

3.4. 통신 구조도 - 순차(시퀀스) 다이어그램

1. Login 및 회원가입 시스템의 시퀀스 다이어그램.



2. 홈 화면의 피드 마커 실시간 생성 및 피드 실시간 수정 시퀀스 DR.



3. 사용자의 로그인 후, 작성된 피드를 열어서 열람할 때 피드 데이터를 전송해주는 시스템의 시퀀스 DR.