라이브러리 관리 서비스

(Team 5: Final Document)

Instructor : Dr. Seok-Won Lee

Course : Software Engineering

Date : 7 December 2023

Team Members :

이정민

김동현

김준하

고종환

Table of Contents

Table of Figures

Table of Tables

1. Motivation and Background

정보통신산업진흥원(NIPA)이 최근 발간한 '2022 오픈소스 SW(OSS) 실태조사 보고서'에 따르면 국내 기업의 오픈소스 활용률은 67.3%인 것으로 나타났다. 조사기업 1188 개 가운데 800 개가 활용하고 있다고 응답했다. 61.5%였던 2021 년과 비교해 약 5.8%P 증가한 수치다. 시장 규모 또한 3380 억원으로 11.4% 성장했다.[1] 이러한 오픈소스 SW 시장의 성장은 개발자 공동체에서 오픈소스 라이브러리 사용이 활발해지고 있다는 것을 의미한다. 따라서 개발자들은 자신들이 사용할 수 있는 다양한 라이브러리에 대해 잘 이해하고 관리하는 것이 중요해졌다.

개발자는 각자 필요에 의해 다양한 라이브러리들을 다양한 버전으로 사용하고 있다. 다양한 라이브러리 중에서도 버전에 따라 사용하는 방식이 다른 경우가 있기 때문에, 개발자가 라이브러리를 선정할 때도 이를 잘 고려하여 결정해야 한다.

라이브러리를 어떤 버전으로 어떻게 사용하는지에 대한 인사이트를 얻기 위해 다른 사람이 진행한 프로젝트에서 어떤 라이브러리를 어떤 방식으로 사용했는지를 검색하는 경우가 잦다. 그리고 코드를 이해하기 위해서는 다른 개발자들의 프로젝트에서 사용된 라이브러리와 코드에 대한 이해가 필수적이다. 그러나 이를 파악하는 과정은 번거롭고 복잡하여 해결 과정이 난해하다.

[1] 권혜미 기자, *기업 10 곳 중 7 곳, 오픈소스 SW 쓴다*, 『전자신문 etnews』, 2023-03-26 17:00, https://www.etnews.com/

2. Project Problem Statement

- **진행했던 프로젝트를 다시 봤을 때 사용했던 기술들을 찾는 소요 시간**

개인이 공부 또는 개발 목적으로 소프트웨어 프로젝트를 진행하고, 이러한 프로젝트에서 사용하는 라이브러리 및 코드 관리의 중요하다.

프로젝트를 볼 때 무슨 라이브러리를 무슨 버전으로 다루었는지, 또는 자료조사 과정에서 프로젝트에 유용하게 사용했던 라이브러리의 사용법이 기억이 나지 않는다면, 이를 복기하기 위해서 진행했던 프로젝트를 저장해둔 Git 이나 File Explorer 를 통해 package.json 이나 gradle 파일을 열람해야 하는 번거로운 작업이 요구된다. 결국 package.json 을 열람한다고 하더라도 단순히 라이브러리의 이름과 버전만 기록되어 있을 뿐, 어떤 메서드나 모듈이 있었고 어떻게 사용했는지 파악하려면 더 깊게 살펴봐야 한다.

- **프로젝트를 타인에게 설명하기 좋은 서비스의 부재**

소프트웨어 역량을 보여주기 위해 개발자는 많은 시간을 코딩, 디버깅, 테스팅 하는데 많은 시간을 프로젝트에 사용한다. 하지만 개발만큼 중요한 것은, 다른 사람에게 자신의 역량을 보여주기 위해 진행했던 프로젝트와 쓰인 기술을 잘 설명하는 일이다. 하지만 정해진 문맥이 없어, 각자 설명하는 방식이 다르기에 어떻게 보여줄 지 고민하고 작성하는 데에 시간이 많이 소요된다.

이러한 고민은 신입 개발자 뿐만 아니라, 프리랜서나 시니어 개발자들이 이직을 할 때나 프리랜서가 이력서를 작성할 때에도 항상 고민되는 부분이다.

- **다른 개발자의 프로젝트 파악의 어려움**

코드는 다양한 기술과 프레임워크를 사용하여 작성된다. 다른 개발자가 사용한 기술이나 라이브러리에 익숙하지 않다면 해당 코드를 이해하기 어렵다.

또한 코드를 작성한 개발자의 의도를 정확하게 파악하기 어려울 때가 있다. 코드를 보고 어떤 문제를 해결하려고 했는지, 어떤 설계 판단을 했는지 이해하기 어려운 경우가 있다.

3. Objective

- **프로젝트 기술 관리의 용이성 향상**

개발자가 진행했던 프로젝트에서 사용한 기술, 라이브러리, 버전을 후에 쉽게 파악할 수 있도록 정리하는 사용자 친화적 웹 서비스를 제공한다.

- **프로젝트 설명 및 문서화의 편의성 증진**

프로젝트를 타인에게 설명하고 문서화 할 때, 효율적이고 일관된 방법 또는 양식을 제공하여 적절한 양식을 고민하여 작성하는 시간과 노력을 절감한다.

- **다른 개발자와의 지식 공유 강화**

웹 서비스에서 프로젝트 검색 기능을 통해 자신이 원하는 분야나 라이브러리에 대해 다른 프로젝트를 검색할 수 있다면, 다른 개발자가 진행한 프로젝트에서 어떤 기술을 어떻게 썼는지 설명된 예시를 통해 코드를 파악하기 용이할 것이다. 따라서 라이브러리나 해시태그를 통한 검색엔진 제공를 제공하면 필요한 라이브러리를 자신의 프로젝트에 적용할 수 있는 기회를 제공하거나, 새로운 라이브러리의 사용 방법을 터득하는 등의 지식 공유를 강화할 수 있다.

4. Proposed Idea / Approach

Libhub 는 개발자들이 자신의 소프트웨어 프로젝트에서 사용된 라이브러리를 효율적으로 관리하고 공유할 수 있도록 돕는 플랫폼이다. 이 시스템은 프로젝트의 라이브러리를 사용자가 수동으로 입력하고 관리할 수 있는 기능을 제공한다. 사용자는 프로젝트의 간단한 설명, 사용된 라이브러리, 버전 정보 등을 기록하고 공유할 수 있다.

- **효율적인 라이브러리 관리 서비스**

사용자는 자신의 프로젝트에서 사용된 라이브러리들을 등록하고 관리할 수 있다. 이를 통해 각 프로젝트에 사용된 라이브러리들을 효과적으로 관리하며, 라이브러리의 이름, 버전, 해시태그, 사용 사례 및 설명을 기록하여 자세한 정보를 제공한다.

- **라이브러리 정보를 공유할 수 있는 서비스**

사용자들은 자신의 프로젝트에 사용된 라이브러리 정보를 공유할 수 있다. 또한, 다른 사용자들의 프로젝트에서 사용된 라이브러리 정보를 조회하며 유용한 인사이트를 얻을 수 있는 라이브러리 레퍼런스 공간을 형성한다. 사용자는 현재 트렌드에 맞는 인기 라이브러리 정보를 쉽게 얻을 수 있으며, 관심 있는 프로젝트들을 저장하여 언제든지 재조회할 수 있다.

5. Scope

5.1 In The Scope

- 시스템은 사용자가 프로젝트 저장소(Git 등)에서 프로젝트 메타데이터를 수동으로 입력하고 관리할 수 있다. 간단한 설명, 라이브러리, 버전 정보를 기록할 수 있다.
- 시스템은 개발자의 소프트웨어 프로젝트를 포트폴리오 형태로 관리하고 다른 이용자와 공유할 수 있는 기능을 제공한다.
- 시스템은 다양한 프로젝트를 검색하고 필터링하여 원하는 라이브러리와 연관된 프로젝트를 찾을 수 있는 기능을 제공한다.

5.2 Out Of Scope

- 시스템은 개발자가 진행한 프로젝트에 중점을 두고 있으며, 개인적인 코드 저장소(예: 개인 블로그, GitHub)의 관리는 다루지 않는다.
- 시스템은 코드 분석 및 기술 추적을 목적으로 하며, 개인 또는 팀 내의 코드 리뷰 도구의 역할을 대체하지는 않는다.
- 시스템은 코드를 분석하고 문서화하는 데 도움을 주지만, 코드 수정 또는 자동화된 코드 변경은 다루지 않는다.
- 시스템은 코드를 실행하거나 테스트하지 않으며, 오직 코드 분석 및 문서화에 관련된 정보를 제공한다.
- 시스템은 서비스 사용자의 코드와 프로젝트 데이터는 개인 정보 보호 및 데이터 보안을 고려하여 안전하게 처리되어야 한다. 서비스 관리자는 사용자 개인의 데이터를 임의로 수정하거나 이용하지 않으며, 서비스 사용자가 자신의 데이터를 직접 관리해야 한다.
- 시스템은 서비스 사용자가 자신의 프로젝트와 기술을 관리하고 공유하는 데에 중점을 둔다.

6. User Requirements

No.	USER_RS_01
Title	사용자는 소셜 로그인으로 회원가입 할 수 있다.
Description	사용자는 자신의 Google 계정으로 서비스에 가입할 수 있다.

Table 1: User Requirements 01

No.	USER_RS_02
Title	사용자는 소셜 로그인으로 서비스에 로그인할 수 있다.
Description	사용자는 서비스에 회원가입했던 Google 계정으로 서비스에 로그인하여 회원 서비스를 이용할 수 있다.

Table 2: User Requirements 02

No.	USER_RS_03
Title	회원은 서비스에서 로그아웃 할 수 있다.
Description	회원은 로그아웃 버튼을 눌러서 서비스에서 로그아웃 할 수 있다.

Table 3: User Requirements 03

No.	USER_RS_04
Title	회원은 서비스에서 회원 탈퇴를 할 수 있다.
Description	회원은 자신이 가입한 아이디로 로그인 한 상태에서 회원 탈퇴를 할 수 있다. 회원 탈퇴하여 생긴 데이터 손실은 복구는 불가능하다.

Table 4: User Requirements 04

No.	USER_RS_05
Title	회원은 자신의 프로필을 수정할 수 있다.
Description	회원은 자신의 이름, 프로필 사진, 자신과 관련된 링크를 수정할 수 있다.

Table 5: User Requirements 05

No.	USER_RS_06
Title	회원은 자신의 프로젝트를 등록할 수 있다.

Description	회원은 프로젝트 이름, 해시태그, 설명, 관련 링크, 공개 여부를 입력한 후 프로젝트를 등록할 수 있다.

Table 6: User Requirements 06

No.	USER_RS_07
Title	회원은 자신이 등록한 프로젝트에 라이브러리를 등록할 수 있다.
Description	회원은 라이브러리의 이름, 해시태그, 버전, 사용 사례를 입력하여 라이브러리를 등록할 수 있다.

Table 7: User Requirements 07

No.	USER_RS_08
Title	회원은 자신이 등록한 프로젝트의 목록을 확인할 수 있다.
Description	회원은 자신이 등록한 프로젝트의 목록을 확인할 수 있다. 각각의 프로젝트 목록에서 해당 프로젝트의 이름, 해시태그, 설명을 확인할 수 있다.

Table 8: User Requirements 08

No.	USER_RS_09
Title	사용자는 읽을 권한이 있는 프로젝트의 상세 내용을 확인할 수 있다.
Description	사용자는 프로젝트 상세 페이지에서 프로젝트 명, 작성자, 해시태그, 설명, 관련 링크, 사용한 라이브러리를 확인할 수 있다. 사용한 라이브러리의 정보에는 라이브러리의 이름, 해시태그, 버전, 사용 사례가 포함된다.

Table 9: User Requirements 09

No.	USER_RS_10
Title	사용자는 자신이 등록한 프로젝트를 수정할 수 있어야 한다.
Description	사용자는 자신이 등록한 프로젝트의 이름, 해시태그, 설명, 관련 링크. 또한 사용한 라이브러리를 추가, 삭제, 수정할 수 있고, 사용한 라이브러리의 이름, 해시태그, 버전, 사용 사례를 수정할 수 있다.

Table 10: User Requirements 10

No.	USER_RS_11
Title	사용자는 자신이 등록한 프로젝트를 삭제할 수 있다.
Description	사용자는 자신이 등록한 프로젝트를 삭제할 수 있다.

Table 11: User Requirements 11

No.	USER_RS_12
Title	사용자는 검색어를 입력하여 프로젝트를 검색하고 검색 결과를 목록으로 확인할 수 있다.
Description	사용자는 프로젝트의 이름, 해시태그를 검색어로 입력하여 프로젝트를 검색하고 검색 결과를 목록으로 확인할 수 있다.

Table 12: User Requirements 12

No.	USER_RS_13
Title	회원은 프로젝트에 좋아요를 등록 및 해제할 수 있다.
Description	회원은 프로젝트에 좋아요 버튼을 눌러서 좋아요 목록에 등록하거나 좋아요 목록에서 해제할 수 있다.

Table 13: User Requirements 13

No.	USER_RS_14
Title	회원은 좋아요를 누른 프로젝트 목록을 확인할 수 있다.
Description	회원은 자신이 좋아요를 누른 모든 프로젝트의 목록을 확인할 수 있다.

Table 14: User Requirements 14

No.	USER_RS_15
Title	회원은 다른 회원을 팔로우 할 수 있다.
Description	회원은 특정 회원의 프로필 페이지나 팔로우 목록에서 팔로우/언팔로우 할 수 있다.

Table 15: User Requirements 15

No.	USER_RS_16
Title	사용자는 다른 회원의 프로필을 확인할 수 있다.
Description	사용자는 다른 회원의 프로필 페이지에서 이름, 프로필 사진, 관련된 링크, 자주 사용하는 라이브러리를 확인할 수 있다.

Table 16: User Requirements 16

No.	USER_RS_17
Title	회원은 팔로워와 자신이 팔로우하는 사용자 목록을 확인할 수 있다.
Description	회원은 자신의 팔로워와 회원이 팔로우 하는 사용자 목록을 각각 확인할 수 있다. 목록에서 각각의 사용자를 조회하여 프로필 조회 페이지로 이동할 수 있고 팔로우 및 언팔로우가 가능하다.

Table 17: User Requirements 17

No.	USER_RS_18
Title	회원은 자신이 팔로우 하는 회원이 등록한 프로젝트 목록을 확인할 수 있다.
Description	회원은 대시보드에서 자신이 팔로우 하는 회원이 등록한 프로젝트 목록을 확인할 수 있다.

Table 18: User Requirements 18

No.	USER_RS_19
Title	사용자는 시스템에서 많이 사용된 라이브러리 목록을 확인할 수 있다
Description	사용자는 많이 사용된 라이브러리 목록을 확인할 수 있다. 많이 사용된 라이브러리는 서비스에 등록된 프로젝트들을 기준으로 삼는다.

Table 19: User Requirements 19

7. System Requirements

7.1 Functional Requirements

No.	SYS_RS_FR_01	**Related Requirements**	USER_RS_01
Title	시스템은 사용자가 소셜 계정으로 회원 등록을 할 수 있도록 한다.		
Description	시스템은 사용자가 가입하려는 Google 계정 실제 존재하는 계정인지, 서비스에 처음 가입하는 계정인지 검사한 후에 서비스에 회원가입 할 수 있도록 한다. 언급한 둘을 제외한 다른 소셜 로그인이나 일반 로그인은 지원하지 않는다.		

Table 20: Functional Requirements 01

No.	SYS_RS_FR_02	**Related Requirements**	USER_RS_02
Title	시스템은 사용자가 가입한 소셜 계정으로 로그인 할 수 있도록 한다.		
Description	시스템은 사용자가 가입했던 소셜 계정으로 로그인 시도하면 가입한 계정인지 검증한 뒤, 가입했던 계정이면 로그인에 성공하도록 한다.		

Table 21: Functional Requirements 02

No.	SYS_RS_FR_03	**Related Requirements**	USER_RS_02
Title	시스템은 로그인에 성공한 회원의 세션을 유지할 수 있어야 한다.		
Description	시스템은 회원가 로그인 하면 세션을 유지하여, 다시 로그인 할 필요가 없도록 해야 한다. 세션이 끝나는 조건은, 회원가 아무 활동이 없어진지 30 일이 지난 뒤거나 다른 브라우저에서 로그인 했을 때이다. 세션 구현 방식은 Session ID 로 한다.		

Table 22: Functional Requirements 03

No.	SYS_RS_FR_04	**Related Requirements**	USER_RS_02
Title	시스템은 세션이 사라지기 전에 서비스를 이용할 경우 세션을 연장한다.		
Description	시스템은 세션이 파괴되는 시간 전에 서버에 요청이 들어올 경우, 세션이 끝나는 시간을 다시 30 일로 설정한다.		

Table 23: Functional Requirements 04

No.	SYS_RS_FR_05	Related Requirements	USER_RS_03
Title	시스템은 회원 로그아웃 할 수 있도록 한다.		
Description	시스템은 회원이 로그아웃 버튼을 누르면 세션을 사라지게 하고 로그아웃을 완료하도록 한다.		

Table 24: Functional Requirements 05

No.	SYS_RS_FR_06	Related Requirements	USER_RS_04
Title	시스템은 회원이 자신의 가입한 계정을 회원 탈퇴할 수 있도록 한다.		
Description	시스템은 로그인 상태인 회원이 자신의 아이디를 회원 탈퇴할 수 있도록 한다. 회원 탈퇴한 계정은 복구가 불가능하고, 자신이 등록했던 프로젝트들은 삭제한다.		

Table 25: Functional Requirements 06

No.	SYS_RS_FR_07	Related Requirements	USER_RS_05
Title	시스템은 회원이 자신의 프로필을 수정할 수 있도록 한다.		
Description	시스템은 회원이 자신의 프로필을 수정할 수 있도록 한다. 수정하는 권한은 회원 자신으로 한정하고, 프로필 이미지의 크기는 10MB 로 제한한다. 사용자 이름,		

Table 26: Functional Requirements 07

No.	SYS_RS_FR_08	Related Requirements	USER_RS_06
Title	시스템은 회원이 프로젝트를 등록할 때 필수로 입력해야 하는 사항이 누락되지 않도록 해야 한다.		
Description	시스템은 회원이 프로젝트를 등록할 때 필수로 입력해야 하는 사항(프로젝트 이름, 공개 여부)의 입력 여부를 확인한다. 필수로 입력해야 하는 사항이 누락 되었다면 회원은 프로젝트 등록에 실패한다.		

Table 27: Functional Requirements 08

No.	SYS_RS_FR_09	Related Requirements	USER_RS_07
Title	시스템은 회원이 라이브러리를 등록할 때 필수로 입력해야 하는 사항이 누락되지 않도록 해야 한다.		
Description	시스템은 회원이 자신이 등록한 프로젝트에 라이브러리를 등록할 때 필수로 입력해야 하는 사항(라이브러리 이름, 버전)의 입력 여부를 확인한다. 필수로 입력해야 하는 사항이 누락 되었다면 회원은 라이브러리 등록에 실패한다.		

Table 28: Functional Requirements 09

No.	SYS_RS_FR_10	Related Requirements	USER_RS_06, USER_RS_07
Title	시스템은 회원이 해시태그를 등록할 때 해당 단어가 해시태그로 등록된다는 사실을 UI 로 나타내야 한다.		
Description	시스템은 회원이 프로젝트나 라이브러리를 등록할 때 해시태그를 입력하면 해당 단어가 특별한 형태로 나타나도록 한다.		

Table 29: Functional Requirements 10

No.	SYS_RS_FR_11	Related Requirements	USER_RS_08, USER_RS_12, USER_RS_14, USER_RS_18
Title	시스템은 회원이 한 번에 볼 수 있는 목록의 수를 제한해야 한다.		
Description	시스템은 회원이 확인할 수 있는 모든 종류의 목록에 대해 페이지네이션을 적용하고 회원에게 페이지에 맞는 목록을 제공해야 한다.		

Table 30: Functional Requirements 11

No.	SYS_RS_FR_12	Related Requirements	USER_RS_09
Title	시스템은 회원 자신의 프로젝트에 대해 수정/삭제 버튼을 제공해야한다.		
Description	시스템은 프로젝트 상세 내용을 확인하는 회원과 해당 프로젝트를 등록한 회원의 일치 여부를 확인해야 한다. 회원의 정보가 일치한다면 프로젝트 상세 페이지에서 프로젝트의 수정, 삭제 버튼을 제공해야 한다.		

Table 31: Functional Requirements 12

No.	SYS_RS_FR_13	Related	USER_RS_10,

		Requirements	USER_RS_11
Title	시스템은 회원 자신의 프로젝트를 수정/삭제할 수 있도록 한다.		
Description	시스템은 프로젝트의 수정 또는 삭제를 시도하는 회원과 프로젝트를 등록한 회원의 일치 여부를 확인해야 한다. 회원의 정보가 일치하지 않는다면 회원은 프로젝트의 수정 또는 삭제에 실패한다.		

Table 32: Functional Requirements 13

No.	SYS_RS_FR_14	**Related Requirements**	USER_RS_12
Title	시스템은 사용자에게 프로젝트 검색 기능을 제공한다.		
Description	시스템은 사용자가 검색어를 입력하면 해당 검색어를 프로젝트의 제목, 프로젝트의 해시태그를 갖는 모든 프로젝트를 사용자에게 목록으로 제공해야 한다.		

Table 33: Functional Requirements 14

No.	SYS_RS_FR_15	**Related Requirements**	USER_RS_13
Title	시스템은 회원에게 프로젝트 좋아요 기능을 제공한다.		
Description	시스템은 회원이 좋아요 버튼을 볼 때 해당 프로젝트에 대한 회원의 좋아요 등록 여부를 알 수 있어야한다. 좋아요를 등록한 상태라면 좋아요 등록 해제 버튼, 좋아요를 등록하지 않은 상태라면 좋아요 등록 버튼이 되어야 한다.		

Table 34: Functional Requirements 15

No.	SYS_RS_FR_16	**Related Requirements**	USER_RS_14
Title	시스템은 회원이 좋아요한 프로젝트 목록을 제공한다.		
Description	시스템은 회원 자신이 좋아요한 프로젝트 목록 제공을 요청하면 회원이 좋아요 등록한 프로젝트 목록을 제공한다.		

Table 35: Functional Requirements 16

No.	SYS_RS_FR_17	**Related Requirements**	USER_RS_15

Title	시스템은 회원이 다른 회원을 팔로우, 팔로우 해제를 할 수 있게 해야 한다.
Description	시스템은 회원이 다른 회원을 팔로우 하는 버튼을 눌렀을 때 해당 회원에 대한 회원의 팔로우 여부를 확인해야 한다. 팔로우 중인 회원이라면 팔로우 해제, 팔로우 중이지 않은 회원이라면 팔로우가 되어야 한다.

Table 36: Functional Requirements 17

No.	SYS_RS_FR_18	**Related Requirements**	USER_RS_16
Title	시스템은 사용자에게 프로필 조회 기능을 제공한다.		
Description	시스템은 사용자가 다른 사용자의 프로필 버튼을 클릭하거나 자신의 프로필을 조회할 때 프로필 정보를 제공하여 사용자가 프로필 정보를 열람할 수 있도록 한다.		

Table 37: Functional Requirements 18

No.	SYS_RS_FR_19	**Related Requirements**	USER_RS_17
Title	시스템은 회원이 팔로우 중인 회원들의 목록을 보여준다		
Description	시스템은 회원이 자신이 팔로우 중인 회원들을 볼 수 있는 페이지를 제공한다.		

Table 38: Functional Requirements 19

No.	SYS_RS_FR_20	**Related Requirements**	USER_RS_18
Title	시스템은 팔로우한 회원의 프로젝트 목록을 보여준다.		
Description	시스템은 회원에게 해당 회원이 팔로우 중인 모든 회원이 등록한 프로젝트 목록을 하나의 대시보드에서 제공한다.		

Table 39: Functional Requirements 20

No.	SYS_RS_FR_21	**Related Requirements**	USER_RS_19
Title	시스템은 회원에게 많이 사용된 라이브러리 목록을 제공해야 한다.		
Description	시스템은 시스템에 등록된 라이브러리를 기준으로 회원에게 많이 사용된 라이브러리 목록을 제공해야 한다. 1 주일의 시작 요일을 월요일로		

	기준으로 하여 해당 주가 시작된 이후 시스템에 등록되어 있는 프로젝트에 등록된 라이브러리의 통계 값이다.

Table 40: Functional Requirements 21

7.2 Non-Functional Requirements

No.	SYS_RS_NFR_01	Related Requirements	
Title	시스템은 평일과 휴일, 주말 상이한 기준 시간 내에 복구돼야 한다.		
Description	시스템은 서버가 다운됐을 때 한국 시간으로 평일(월-금 9 시 - 18 시)에는 3 시간 이내에 이용 가능해야 하고, 이외의 시간이나 한국 기준 휴일, 주말에는 6 시간 내에 이용 가능해야 한다.		

Table 41: Non-Functional Requirements 01

No.	SYS_RS_NFR_02	Related Requirements	
Title	시스템은 모든 페이지의 렌더링 완료되는 시간을 3 초 이내로 한다.		
Description	시스템은 페이지의 모든 정보를 확인할 수 있고 상호작용할 수 있는 시간을 3 초 이내로 한다. 3 초를 넘어가면 에러 페이지를 띄운다.		

Table 42: Non-Functional Requirements 02

No.	SYS_RS_NFR_03	Related Requirements	
Title	시스템은 동시 접속자 수를 최대 40 명을 감당할 수 있어야 한다.		
Description	동시 접속자는 웹 서버에 커넥션하고 있는 사용자 수이다. 동시 접속자 수가 40 명 이하일 때는 서버가 다운되지 않아야 한다.		

Table 43: Non-Functional Requirements 03

No.	SYS_RS_NFR_04	Related Requirements	
Title	시스템은 24 시간 이용 가능해야 한다.		
Description	시스템은 점검시간을 제외하고 24 시간동안 사용자가 접속할 수 있는 상태여야 한다.		

Table 44: Non-Functional Requirements 04

8. Domain Requirements

No.	DM_RS_01	Related Requirements	USER_RS_01 SYS_RS_FR_01
Title	GDPR 및 CCPA 를 준수한다.		
Description	시스템은 사용자의 개인정보를 다루는 경우에 GDPR(유럽 연한 일반 데이터 보호 규칙) 및 CCPA(캘리포니아 정보 보호 법)을 위반하는 경우가 없어야 한다.		

Table 45: Domain Requirements 01

9. Supplementary Requirements

9.1 Interface requirements

- 웹 페이지를 통해 서비스를 운영한다. 유저 인터페이스는 메인 페이지(많이 사용한 라이브러리, 대시보드), 내가 등록한 프로젝트 페이지, 팔로워/팔로잉 목록 페이지, 프로필 관리 페이지로 총 5 가지 메뉴로 구성된다.

9.2 Physical requirements

- 시스템은 모든 페이지의 렌더링 완료되는 시간을 3 초 이내로 한다.
- 서버는 24 시간 이용 가능해야 하고, 최대 50 명의 동시접속자를 감당할 수 있어야한다.
- 서버가 다운되는 경우 평일 3 시간 이내, 주말 6 시간 이내에 복구할 수 있어야 한다.

9.3 Design requirements

- SPA 기반의 React 로 프론트 엔드를 구성하고, JAVA 를 사용하는 Spring 으로 백엔드 서버를 구성한다.
- Database 는 정교한 RDB 가 필요한 데이터의 경우 MySQL 을 활용하고, 프로젝트 등록과 같은 용량이 큰 데이터는 Amazon S3 에 저장한다.

9.4 Implementation requirements

- Frontend - React
- Backend - Spring Boot, JPA
- Database - MySQL

9.5 Security

- 비회원 사용자는 프로젝트 등록, 수정, 관심 프로젝트 저장 기능을 이용할 수 없고 사용자들은 다른 사용자들의 프로젝트에 대해 수정 권한이 없다. 또한, 비공개로 설정된 프로젝트는 다른 사용자들에게 노출 되지 않는다.

10. Use Case

10.1 Use Case Diagram

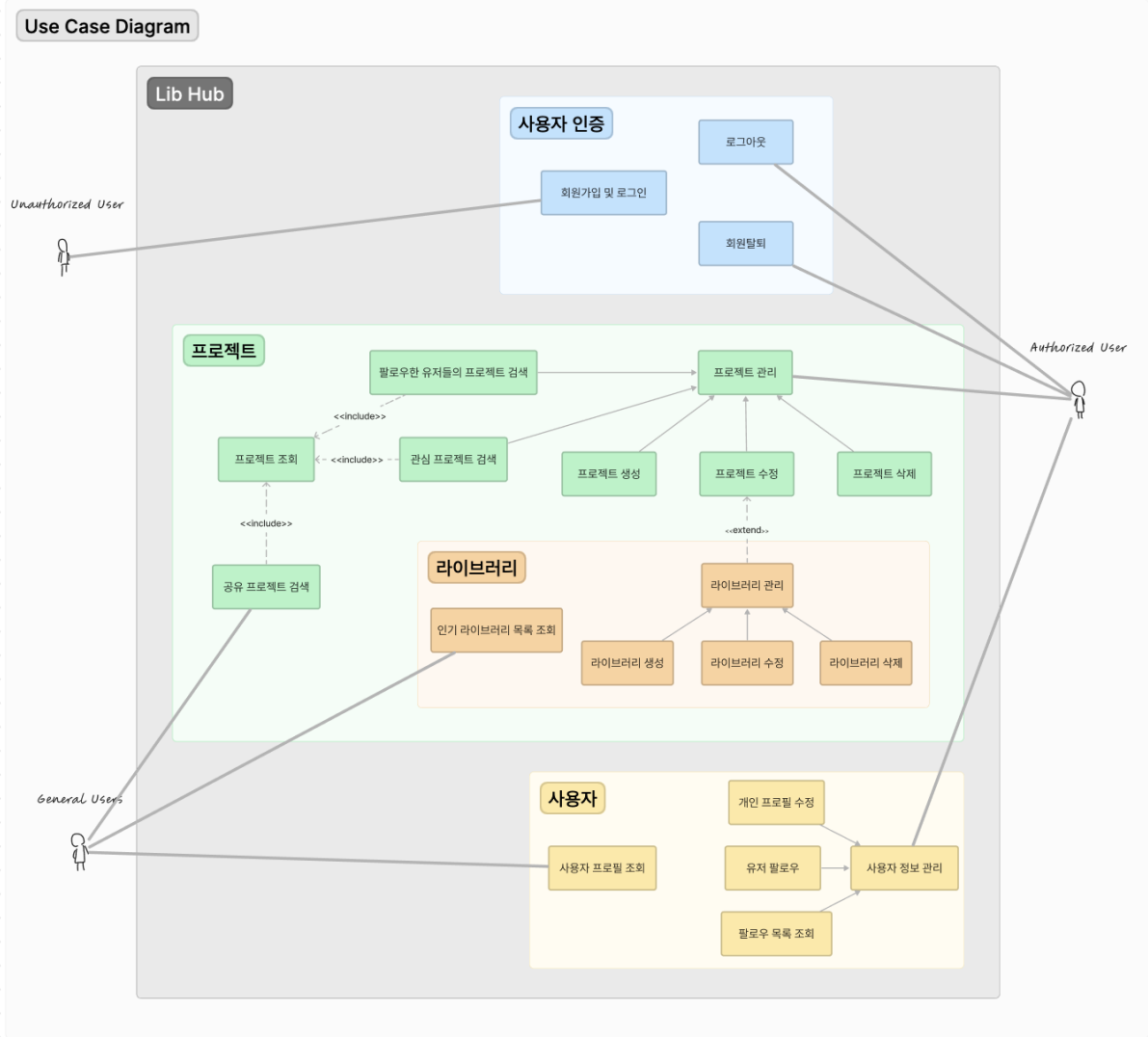


Figure 01: Use Case Diagram

10.2 Use Case Specification

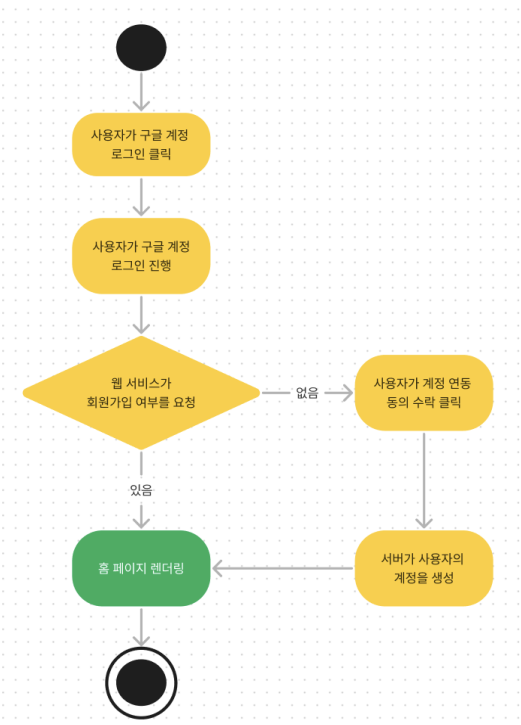
Use-Case ID	1
Use-Case Name	로그인/회원가입
Actors	사용자, 웹 서비스, 서버
Description	사용자가 구글 계정과 연동하여 웹 서비스에 회원 가입하고 로그인 한다.
Pre-conditions	사용자는 웹 서비스에 로그인 되어있지 않아야 한다. 사용자는 로그인 페이지에 접속해 있어야 한다.
Post-conditions	사용자는 회원 가입을 했다면 서버에 계정이 저장된다. 사용자는 웹 서비스에 로그인하고, 회원만의 기능을 이용할 수 있다.
Primary Flow	1. 사용자가 구글 계정 로그인 버튼을 클릭한다. 2. 사용자가 구글 계정으로 로그인을 진행한다. 3. 웹 서비스가 서버에 회원 가입 여부를 요청한다. 4. 서버에 사용자의 계정이 있다면 서버가 회원가입한 사용자의 계정을 생성한다. 5. 웹 서비스가 홈 페이지를 렌더링 한다.
Alternate Flow	4.1. 서버에 사용자의 계정이 없는 경우 1. 사용자는 사용자 계정 연동 동의 수락을 클릭한다. 2. 서버가 회원가입한 사용자의 계정을 생성한다. 3. 5 단계로 이동한다.
Activity Diagram	

Figure 02: Use Case 01 Activity Diagram

Table 46: Use Case 01

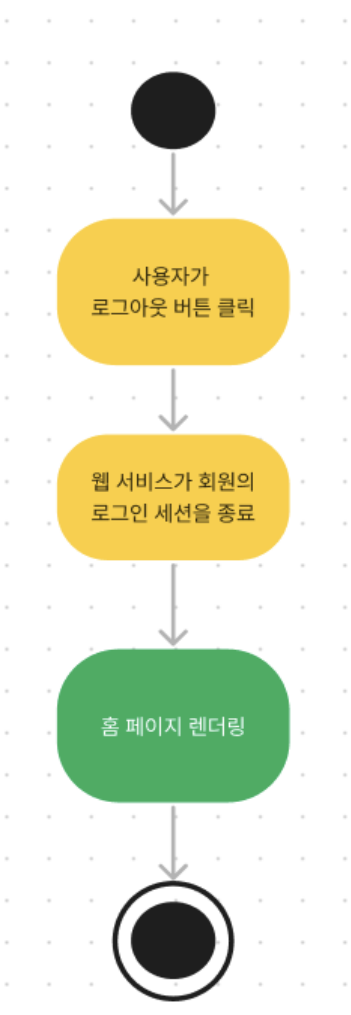
Use-Case ID	2
Use-Case Name	로그아웃
Actors	회원, 웹 서비스, 서버
Description	회원은 웹 서비스에 로그아웃하여 개인 로그인 세션을 종료한다.
Pre-conditions	회원은 웹 서비스에 로그인 상태여야 한다. 최상단 프레임이 렌더링 되어 있어야 한다.
Post-conditions	회원은 로그인 세션이 종료되고 홈페이지로 이동한다.
Primary Flow	1. 회원은 로그아웃 버튼을 클릭한다. 2. 웹 서비스는 회원의 로그인 세션을 종료한다. 3. 웹 서비스는 홈 페이지를 렌더링 한다.
Alternate Flow	없음
Activity Diagram	

	Figure 03: Use Case 02 Activity Diagram

Table 47: Use Case 02

Use-Case ID	3
Use-Case Name	회원 탈퇴
Actors	회원, 웹 서비스, 서버
Description	회원은 회원탈퇴하여 웹 서비스에 계정을 삭제하고 로그인 세션을 종료한다.
Pre-conditions	회원은 웹 서비스에 로그인 상태여야 한다. 웹 서비스는 마이페이지가 렌더링 되어 있어야 한다.
Post-conditions	서버에 회원의 계정이 삭제 된다. 회원은 로그인 세션이 종료되며, 비회원의 기능만 이용할 수 있다.
Primary Flow	1. 회원은 회원탈퇴 버튼을 클릭한다. 2. 웹 서비스는 회원에게 회원탈퇴 재확인을 요청한다. 3. 회원이 확인을 클릭했다면 서버는 회원의 계정을 삭제한다. 4. 웹 서비스는 회원의 로그인 세션을 종료한다. 5. 웹 서비스는 홈 페이지를 렌더링 한다.
Alternate Flow	3.1 회원이 취소를 클릭한 경우 1. 웹 서비스가 마이페이지를 렌더링 한다.

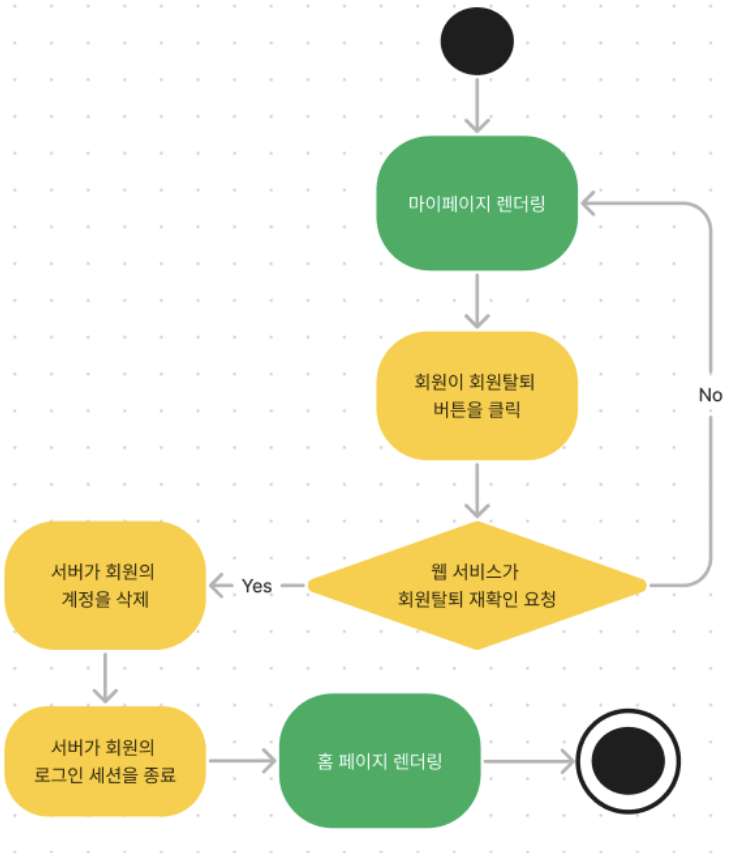
Activity Diagram	

Figure 04: Use Case 03 Activity Diagram

Table 48: Use Case 03

Use-Case ID	4
Use-Case Name	프로젝트 생성
Actors	회원, 웹 서비스, 서버
Description	회원은 웹 서비스에 프로젝트 정보를 입력하여 프로젝트를 생성한다.
Pre-conditions	회원은 웹 서비스에 로그인 상태여야 한다. 웹 서비스는 프로젝트 등록 페이지가 렌더링 되어 있어야 한다.
Post-conditions	서버에 새 프로젝트가 저장된다. 웹 서비스는 프로젝트 상세페이지를 렌더링한다.
Primary Flow	1. 회원은 새 프로젝트 생성 버튼을 클릭한다. 2. 웹 서비스는 프로젝트 정보 입력페이지를 렌더링 한다. 3. 회원은 프로젝트 정보를 입력한다. 4. 회원은 프로젝트 생성 버튼을 클릭한다. 5. 웹 서비스는 회원이 입력한 프로젝트 정보가 유효한 정보인지 확인한다. 6. 웹 서비스가 검증에 성공하면 서버는 새 프로젝트를 저장한다. 7. 웹 서비스는 프로젝트 상세페이지를 렌더링 한다..

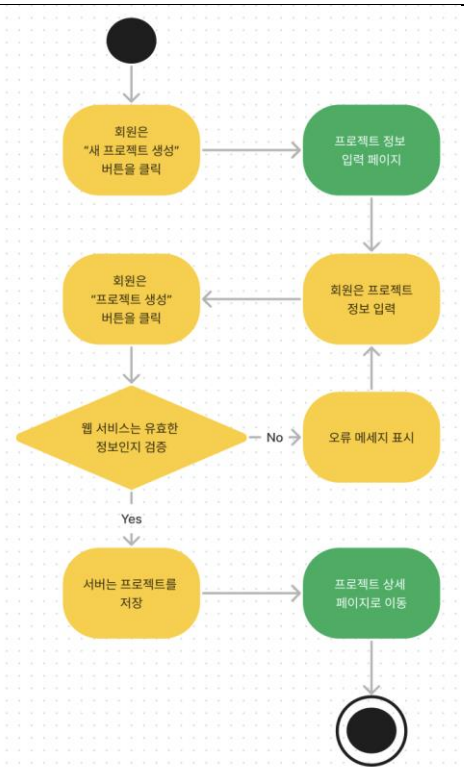
Alternate Flow	5.1 웹 서비스는 검증에 실패한 경우 1. 오류 메시지를 회원에게 표시한다. 2. 3 단계로 이동한다.
Activity Diagram	

Figure 05: Use Case 04 Activity Diagram

Table 49: Use Case 04

Use-Case ID	5
Use-Case Name	프로젝트 조회
Actors	회원, 웹 서비스, 서버
Description	회원은 웹 서비스에 등록된 프로젝트의 상세 정보를 조회한다.
Pre-conditions	회원은 웹 서비스에 로그인 상태여야 한다. 웹 서비스는 프로젝트 목록 페이지가 렌더링 되어 있어야 한다.
Post-conditions	웹 서비스는 프로젝트 상세페이지를 렌더링한다.
Primary Flow	1. 사용자가 프로젝트 목록에서 프로젝트를 선택한다. 2. 웹 서비스가 서버에 상세 정보를 요청한다. 3. 서버가 프로젝트 상세 정보를 제공한다. 4. 웹 서비스는 프로젝트 상세 페이지를 렌더링한다. 5. 사용자가 라이브러리를 선택한다. 6. 웹 서비스는 라이브러리 상세 토글을 렌더링한다.

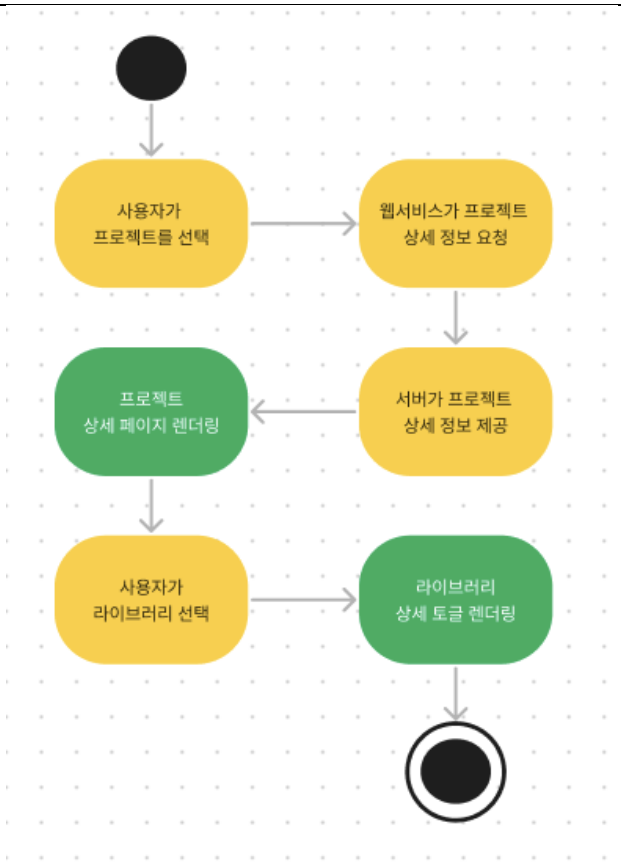
Alternate Flow	없음
Activity Diagram	

Figure 06: Use Case 05 Activity Diagram

Table 50: Use Case 05

Use-Case ID	6
Use-Case Name	프로젝트 수정
Actors	회원, 웹 서비스, 서버
Description	회원은 웹 서비스에 등록된 프로젝트 정보를 수정하거나 삭제한다.
Pre-conditions	회원은 웹 서비스에 로그인 상태여야한다. 수정할 프로젝트가 시스템에 등록되어 있어야한다. 회원은 프로젝트 수정 권한을 가지고 있어야한다.
Post-conditions	서버에 프로젝트의 상세 정보가 수정되거나 삭제된다.

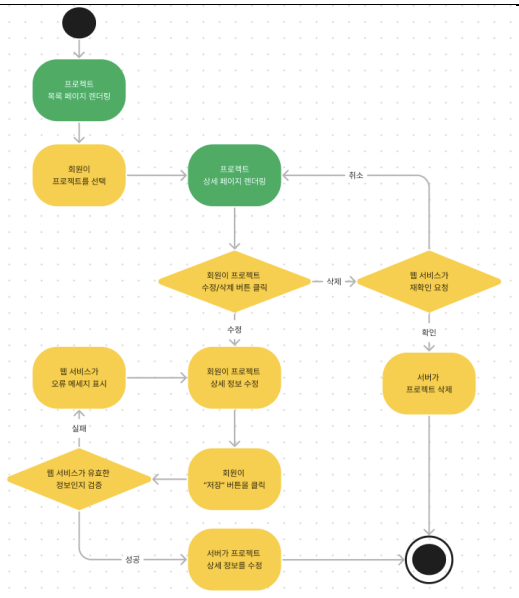
Primary Flow	1. 웹 서비스가 프로젝트 목록 페이지를 렌더링한다. 2. 회원이 프로젝트를 선택한다. 3. 웹 서비스가 프로젝트 상세 페이지를 렌더링한다. 4. 회원이 수정, 삭제 버튼을 클릭한다. 4.1. 삭제 버튼을 클릭했다면 웹 서비스가 프로젝트를 삭제 재확인을 회원에게 요청한다. 4.1.1. 확인을 클릭했다면 서버가 해당 프로젝트를 삭제한다. 4.2. 수정 버튼을 클릭했다면회원이 프로젝트 상세 정보를 수정한다. 4.3. 회원이 “저장” 버튼을 클릭한다. 4.4. 웹 서비스가 회원이 수정한 정보가 유효한지 검증한다. 4.5. 검증에 성공했다면 서버가 프로젝트 상세 정보를 수정한다.
Alternate Flow	4.1.1 취소를 클릭한 경우 1. 웹 서비스가 프로젝트 상세 페이지를 렌더링 한다. 4.4 검증에 실패한 경우 1. 웹 서비스가 오류 메세지를 표시한다 2. 4.2 단계로 이동한다.
Activity Diagram	

Figure 07: Use Case 06 Activity Diagram

Table 51: Use Case 06

Use-Case ID	7
Use-Case Name	라이브러리 생성
Actors	회원, 웹 서비스, 서버
Description	회원은 웹 서비스에 라이브러리 정보를 입력하여 라이브러리를 생성한다.
Pre-conditions	회원은 웹 서비스에 로그인 상태여야 한다. 웹 서비스는 프로젝트 상세 페이지가 렌더링 되어 있어야 한다.

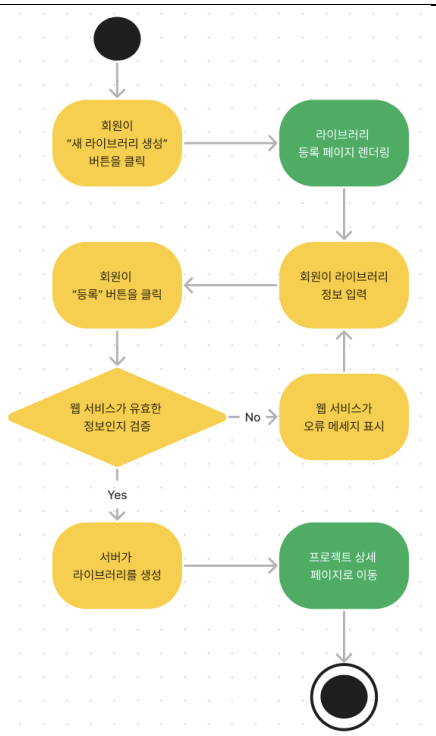
Post-conditions	서버에 프로젝트 내의 새 라이브러리가 저장된다. 웹 서비스는 프로젝트 상세페이지를 렌더링한다.
Primary Flow	1. 회원이 새 라이브러리 생성 버튼을 클릭한다. 2. 웹 서비스가 라이브러리 등록 페이지를 렌더링 한다. 3. 회원이 라이브러리 정보를 입력한다. 4. 입력이 끝나면 회원이 라이브러리 “등록" 버튼을 클릭한다. 5 웹 서비스가 회원이 입력한 라이브러리 정보가 유효한 정보인지 확인한다. 6. 웹 서비스가 검증에 성공하면 서버는 프로젝트에 새 라이브러리를 생성한다. 7. 웹 서비스가 프로젝트 상세페이지를 렌더링 한다.
Alternate Flow	5.1 웹 서비스가 검증에 실패한 경우 1. 오류 메시지를 회원에게 표시한다.
Activity Diagram	

Figure 08: Use Case 07 Activity Diagram

Table 52: Use Case 07

Use-Case ID	8
Use-Case Name	라이브러리 수정
Actors	회원, 웹 서비스, 서버
Description	회원은 웹 서비스에 등록된 라이브러리 정보를 수정하거나 삭제한다.

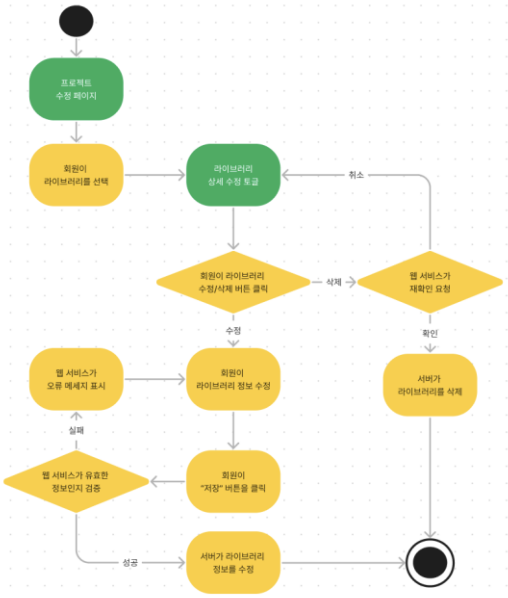
Pre-conditions	회원은 웹 서비스에 로그인 상태여야 한다. 수정할 프로젝트와 라이브러리가 시스템에 등록되어 있어야 한다. 회원은 프로젝트 수정 권한을 가지고 있어야 한다.
Post-conditions	서버에 라이브러리의 상세 정보가 수정되거나 삭제된다.
Primary Flow	1. 웹 서비스가 프로젝트 수정 페이지를 렌더링 한다. 2. 회원이 라이브러리를 선택한다. 3. 웹 서비스가 라이브러리 상세 수정 토글을 렌더링 한다. 4. 회원이 수정, 삭제 버튼을 클릭한다. 5. 삭제 버튼을 클릭했다면 웹 서비스가 라이브러리 삭제 재확인을 회원에게 요청한다. 6. 확인을 클릭했다면 서버가 해당 라이브러리를 삭제한다. 7. 수정 버튼을 클릭했다면 회원이 라이브러리 상세 정보를 수정한다. 8. 회원이 “저장” 버튼을 클릭한다. 9. 웹 서비스가 회원이 수정한 정보가 유효한지 검증한다. 10. 검증에 성공했다면 서버가 라이브러리 상세 정보를 수정한다.
Alternate Flow	5.1 취소를 클릭한 경우 1. 웹 서비스가 라이브러리 상세 수정 토글을 렌더링 한다. 9.1 검증에 실패한 경우 1. 웹 서비스가 오류 메세지를 표시한다. 2. 7 단계로 이동한다.
Activity Diagram	 **Figure 09: Use Case 08 Activity Diagram**

Table 53: Use Case 08

Use-Case ID	9
Use-Case Name	공유 프로젝트 검색

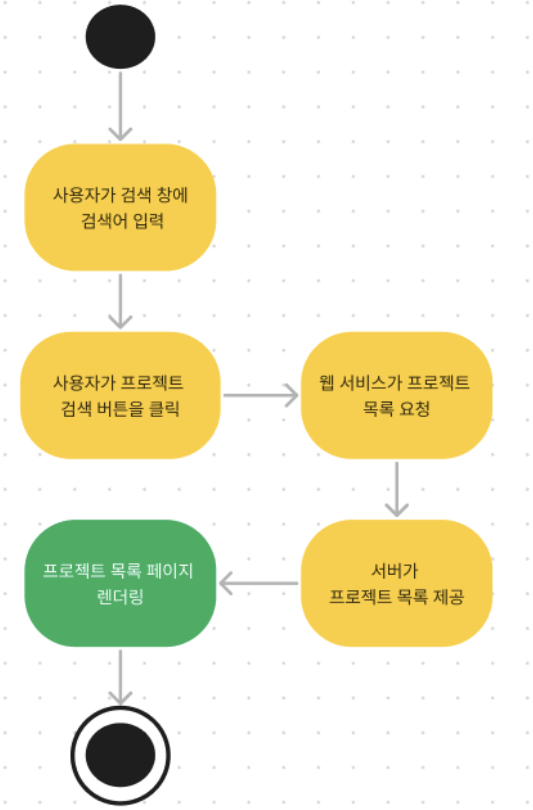
Actors	사용자, 웹 서비스, 서버
Description	사용자는 웹 서비스에 공유된 프로젝트를 검색어를 사용하여 검색한다.
Pre-conditions	검색 창이 렌더링 되어 있어야 한다.
Post-conditions	웹 서비스는 검색어와 일치하는 프로젝트 목록 페이지를 렌더링한다.
Primary Flow	1. 사용자가 검색 창에 검색어를 입력한다. 2. 입력이 끝나면 사용자가 프로젝트 검색 버튼을 클릭한다. 3. 웹 서비스가 검색어와 일치하는 프로젝트 목록을 요청한다. 4. 서버는 검색어와 일치하는 프로젝트들만 제공한다. 5. 웹 서비스가 프로젝트 목록 페이지를 렌더링 한다.
Alternate Flow	없음
Activity Diagram	 **Figure 10: Use Case 09 Activity Diagram**

Table 54: Use Case 09

Use-Case ID	10
Use-Case Name	사용자 프로필 조회

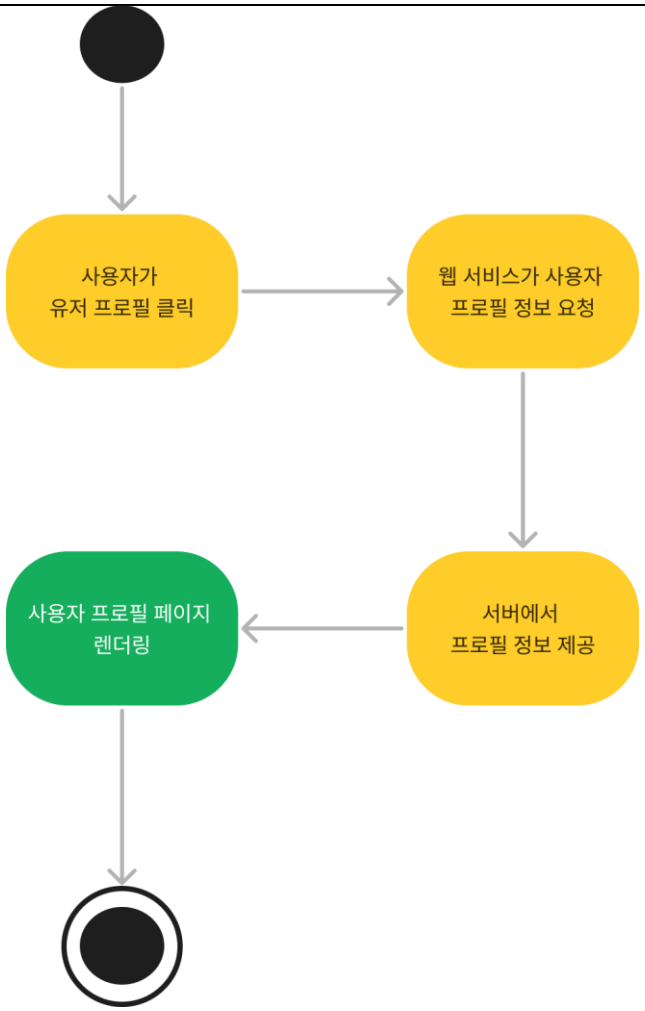
Actors	비회원, 웹 서비스, 서버
Description	사용자가 특정 사용자의 프로필을 클릭하여 프로필 정보를 조회한다.
Pre-conditions	웹 서비스에서 다른 사용자의 프로필 아이콘이 렌더링 된 상태여야 한다.
Post-conditions	사용자가 다른 사용자의 프로필 정보를 볼 수 있다.
Primary Flow	1. 사용자가 다른 사용자의 프로필을 클릭한다. 2. 웹 서비스가 사용자 프로필 정보를 요청한다. 3. 서버에서 프로필 정보를 제공한다. 4. 웹 서비스가 사용자 프로필 페이지를 렌더링한다.
Alternate Flow	없음
Activity Diagram	

Figure 11: Use Case 10 Activity Diagram

Table 55: Use Case 10

Use-Case ID	11
Use-Case Name	관심 프로젝트 조회

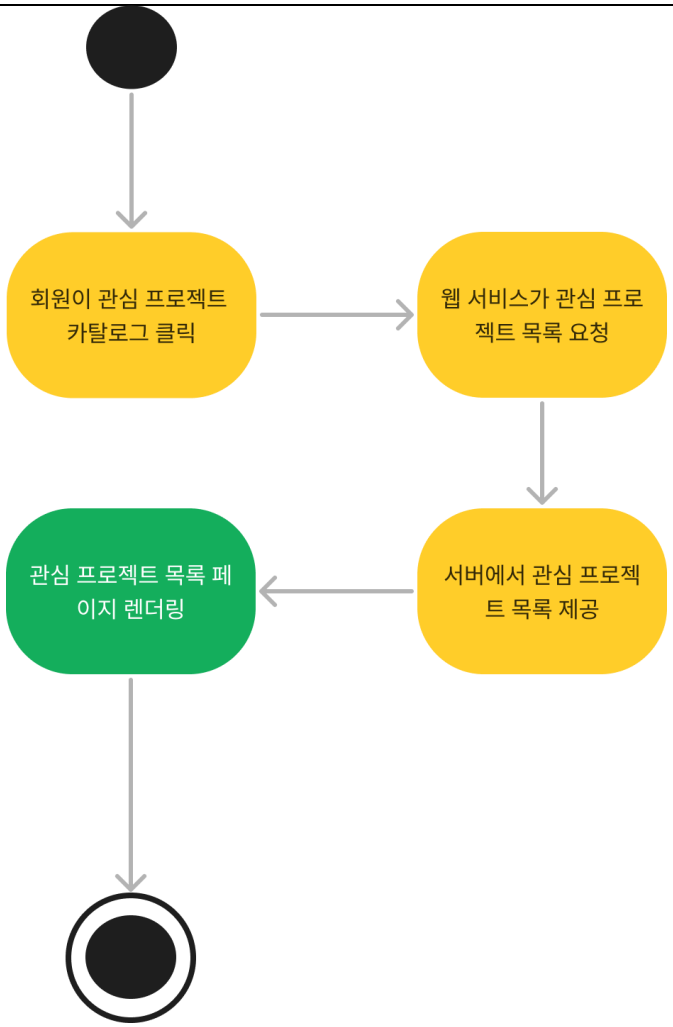
Actors	회원, 웹 서비스, 서버
Description	회원이 “관심 프로젝트” 카탈로그를 클릭하여 자신이 ‘좋아요’한 프로젝트 목록을 조회한다.
Pre-conditions	사용자는 로그인된 회원이어야 한다. 웹 서비스에서 상단 카탈로그가 렌더링 되어 있어야 한다.
Post-conditions	회원이 ‘좋아요’로 지정한 프로젝트 목록을 조회할 수 있다.
Primary Flow	1. 회원이 카탈로그에서 관심 프로젝트 항목을 클릭한다. 2. 웹 서비스가 관심 프로젝트 목록을 서버에 요청한다. 3. 서버에서 회원이 관심 등록한 프로젝트 목록을 제공한다. 4. 웹 서비스가 관심 프로젝트 목록 페이지를 렌더링한다.
Alternate Flow	없음
Activity Diagram	

Figure 12: Use Case 11 Activity Diagram

Table 56: Use Case 11

Use-Case ID	12
Use-Case Name	개인 프로필 수정

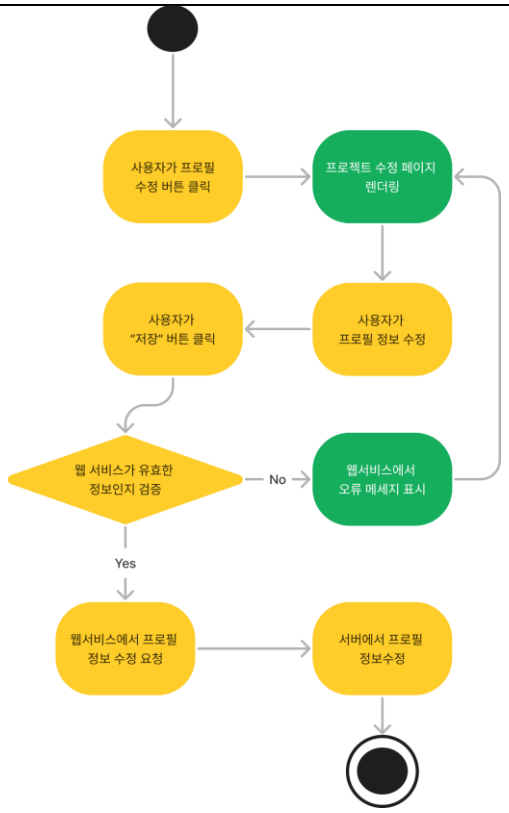
Actors	회원, 웹 서비스, 서버
Description	회원이 개인 프로필 정보를 수정한다. 프로필 정보에서 이름, 프로필 사진, 관련 링크를 수정할 수 있다.
Pre-conditions	사용자는 로그인된 회원이어야 한다. 회원 자신의 프로필 페이지가 렌더링 되어 있어야 한다.
Post-conditions	회원이 입력한 정보로 프로필 정보가 수정된다.
Primary Flow	1. 사용자가 프로필 수정 버튼을 클릭한다. 2. 웹서비스가 프로필 수정 페이지 렌더링한다. 3. 사용자가 프로필 정보를 수정한다. 4. 사용자가 “저장” 버튼을 클릭한다. 5. 웹서비스에서 사용자가 입력한 정보가 유효한지 검증한다. 6. 유효한 경우, 웹서비스가 프로필 정보 수정을 요청한다. 7. 서버가 프로필 정보 수정을 반영한다.
Alternate Flow	5.1 사용자가 입력한 프로필 정보가 유효하지 않는 경우 1. 웹 서비스에서 오류 메시지를 표시한다. 2. 2 단계로 이동한다.
Activity Diagram	

Figure 13: Use Case 12 Activity Diagram

Table 57: Use Case 12

Use-Case ID	13
Use-Case Name	인기 라이브러리 목록 조회

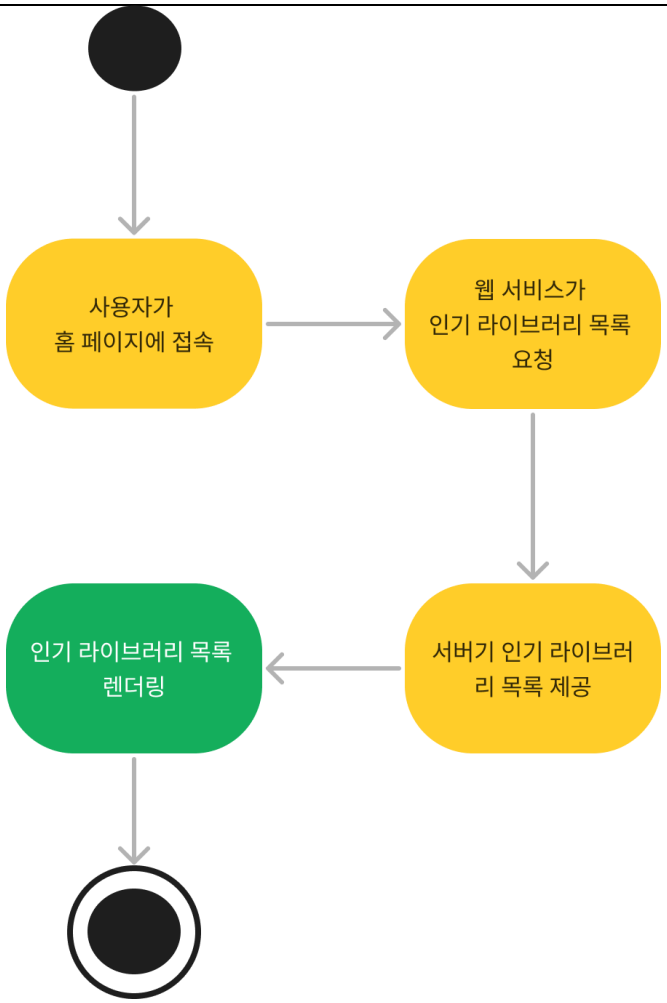
Actors	비회원, 웹 서비스, 서버
Description	사용자가 일주일 간 가장 인기 있는 라이브러리의 목록을 홈 페이지에서 조회한다.
Pre-conditions	사용자가 홈 페이지에 접속 가능한 상태여야 한다.
Post-conditions	사용자가 인기 라이브러리 목록을 조회할 수 있다.
Primary Flow	1. 사용자가 홈 페이지에 접속한다. 2. 웹 서비스가 인기 라이브러리 목록을 요청한다. 3. 서버가 인기 라이브러리 목록을 제공한다. 4. 웹 서비스에서 인기 라이브러리 목록을 렌더링한다.
Alternate Flow	없음
Activity Diagram	

Figure 14: Use Case 13 Activity Diagram

Table 58: Use Case 13

Use-Case ID	14
Use-Case Name	대시보드 조회

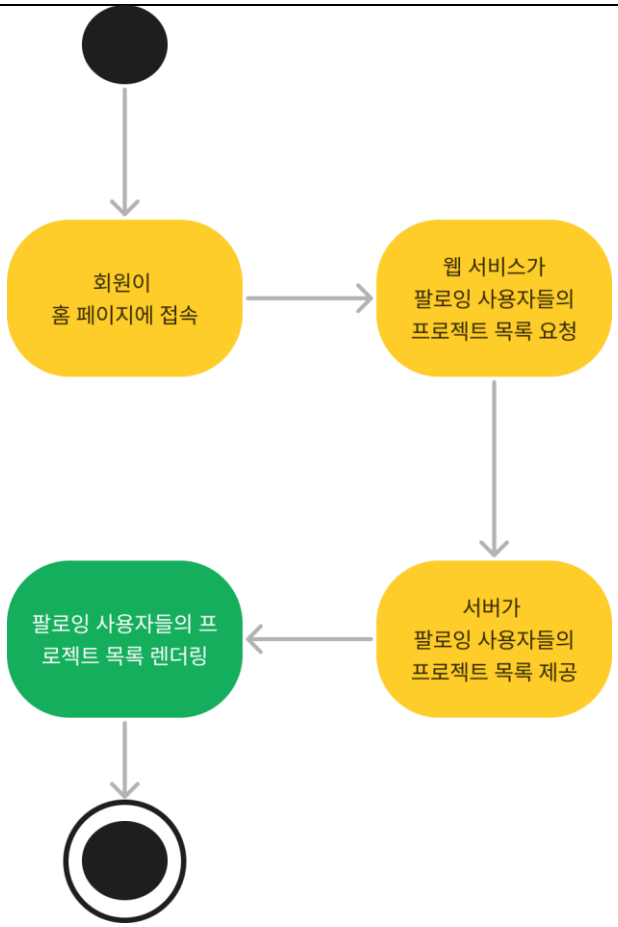
Actors	회원, 웹 서비스, 서버
Description	회원이 팔로우한 사용자들의 프로젝트 목록을 홈 화면에서 볼 수 있다.
Pre-conditions	회원이 홈페이지에 접속 가능한 상태여야 한다.
Post-conditions	회원이 팔로우한 사용자들의 프로젝트 목록을 조회할 수 있다.
Primary Flow	1. 회원이 홈페이지에 접속한다. 2. 웹 서비스가 팔로잉 사용자들의 프로젝트 목록을 요청한다. 3. 서버가 팔로우한 사용자들의 프로젝트 목록을 제공한다. 4. 웹 서비스가 팔로우한 사용자들의 프로젝트 목록을 렌더링한다.
Alternate Flow	없음
Activity Diagram	

Figure 15: Use Case 14 Activity Diagram

Table 59: Use Case 14

Use-Case ID	15
Use-Case Name	팔로잉 목록 조회

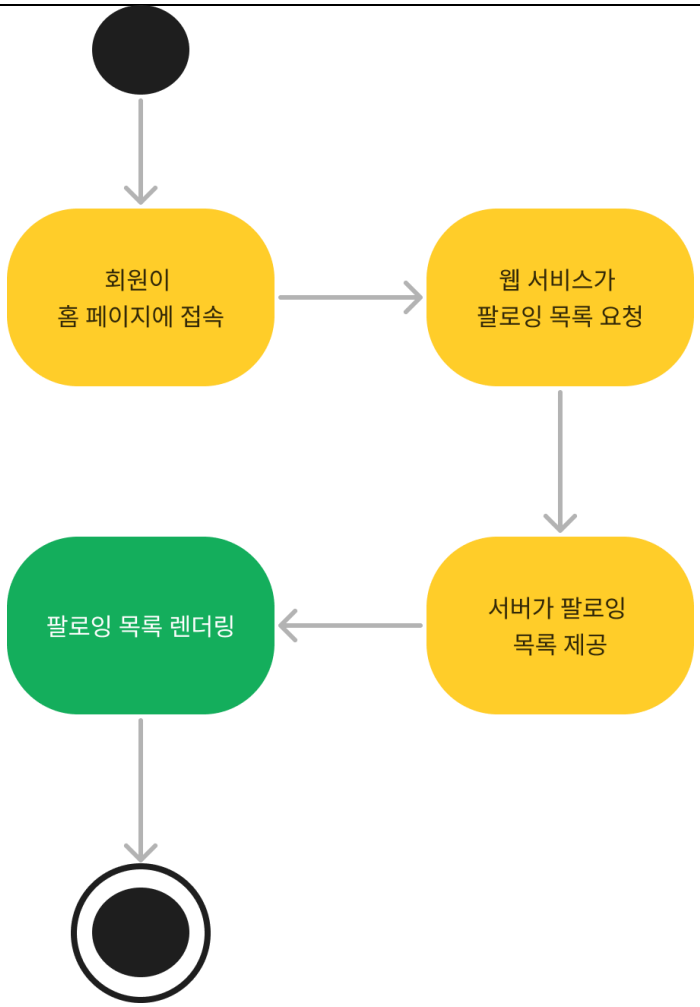
Actors	회원, 웹 서비스, 서버
Description	회원이 홈 페이지에서 자신의 팔로잉 목록을 간편하게 조회할 수 있다.
Pre-conditions	사용자는 로그인 된 회원이어야 한다. 회원이 홈페이지에 접속 가능해야 한다.
Post-conditions	회원의 팔로잉 목록을 조회한다.
Primary Flow	1. 회원이 팔로잉 목록을 클릭한다. 2. 웹 서비스가 팔로잉 목록을 요청한다. 3. 서버가 해당 사용자의 팔로잉 목록을 제공한다. 4. 웹서비스가 홈에서 팔로잉 목록을 렌더링한다.
Alternate Flow	없음
Activity Diagram	

Figure 16: Use Case 15 Activity Diagram

Table 60: Use Case 15

Use-Case ID	16
Use-Case Name	팔로워 목록 조회

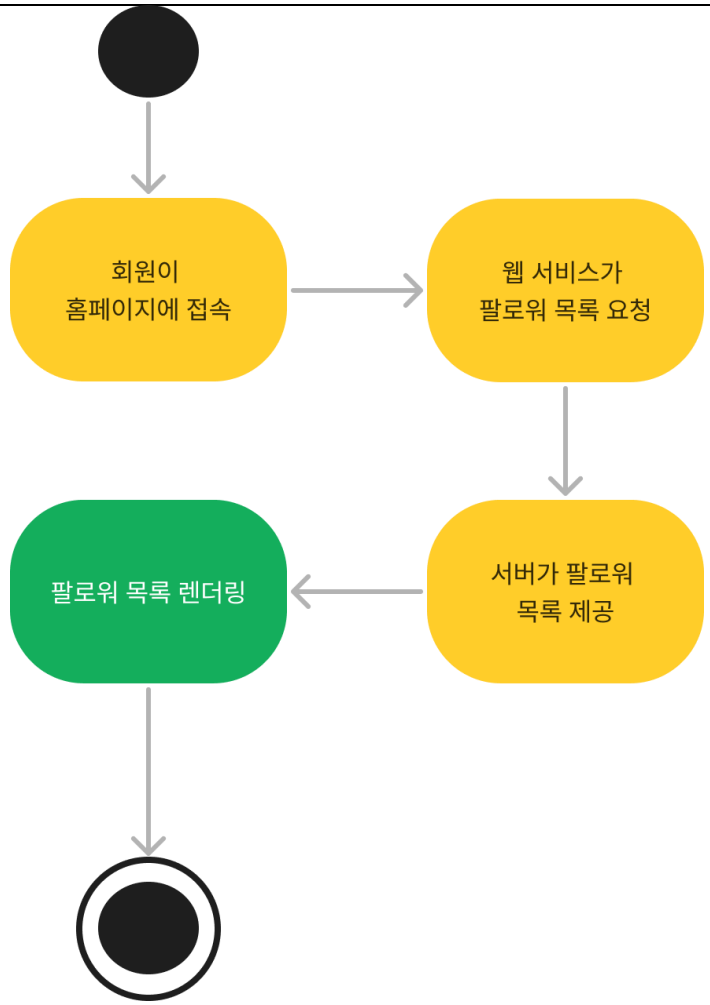
Actors	회원, 웹 서비스, 서버
Description	회원이 홈 페이지에서 자신의 팔로워 목록을 간편하게 조회할 수 있다.
Pre-conditions	사용자는 로그인된 회원어야 한다. 회원은 홈페이지에 접속 가능해야 한다.
Post-conditions	회원이 자신의 팔로워 목록을 조회한다.
Primary Flow	1. 회원이 홈 페이지에 접속한다. 2. 웹 서비스가 팔로워 목록을 요청한다. 3. 서버에서 해당 사용자의 팔로워 목록을 제공한다. 4. 웹 서비스가 홈에서 팔로워 목록을 렌더링한다.
Alternate Flow	없음
Activity Diagram	

Figure 17: Use Case 16 Activity Diagram

Table 61: Use Case 16

Use-Case ID	17
Use-Case Name	팔로우 및 언팔로우

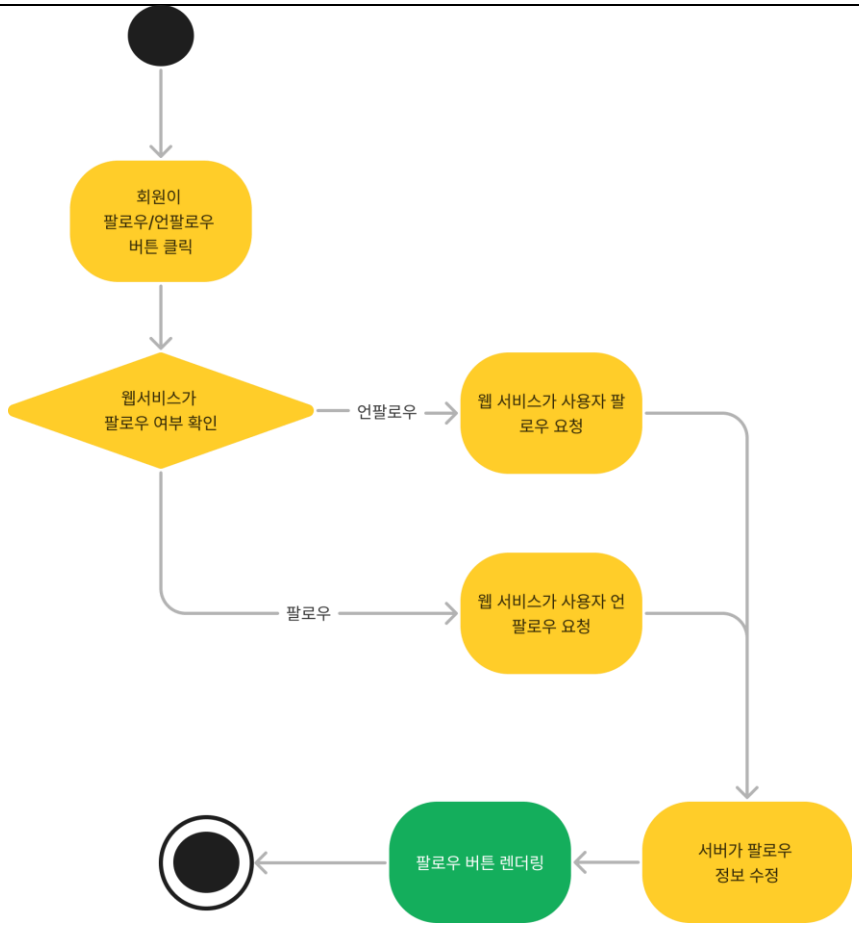
Actors	회원, 웹 서비스, 서버
Description	회원이 다른 사용자의 프로필에서 팔로우 버튼을 누르면 팔로우 되거나 언팔로우 된다.
Pre-conditions	사용자는 로그인된 회원이어야 한다. 회원이 다른 사용자의 프로필 페이지에 있어야 한다.
Post-conditions	회원의 특정 사용자에 대한 팔로우 상태가 토글된다.
Primary Flow	1. 회원이 팔로우/언팔로우 버튼을 클릭한다. 2. 웹 서비스가 팔로우 여부를 확인한다. 3. 언팔로우 상태면, 웹 서비스가 선택된 사용자 팔로우 요청을 보낸다. 4. 서버에서 팔로우/언팔로우 정보를 수정한다. 5. 웹 서비스에서 팔로우 버튼을 렌더링한다.
Alternate Flow	3.1. 팔로우 상태인 경우 1. 웹 서비스가 사용자 언팔로우 요청을 보낸다. 2. 4 단계로 이동한다.
Activity Diagram	

Figure 18: Use Case 17 Activity Diagram

Table 62: Use Case 17

Use-Case ID	18
Use-Case Name	프로젝트 “좋아요” 등록

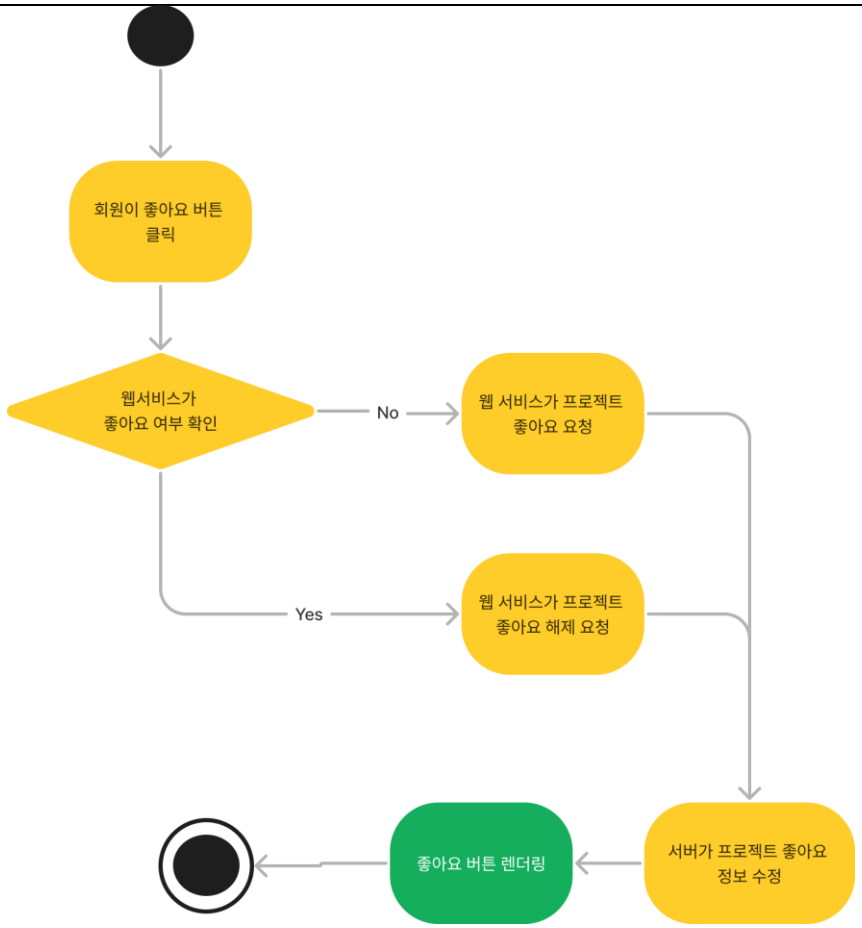
Actors	회원, 웹 서비스, 서버
Description	회원이 다른 사용자의 프로젝트에 좋아요 등록을 하거나 해제한다.
Pre-conditions	사용자는 로그인된 회원이어야 한다. 회원이 다른 사용자의 프로젝트 상세 페이지에 있어야 한다.
Post-conditions	회원의 특정 프로젝트에 대한 “좋아요” 상태가 토글된다.
Primary Flow	1. 회원이 “좋아요” 버튼을 클릭한다. 2. 웹 서비스가 해당 프로젝트의 “좋아요” 여부를 확인한다. 3. “좋아요” 상태가 아니면, 웹 서비스가 선택된 프로젝트를 “좋아요” 등록 요청을 보낸다. 4. 서버에서 프로젝트 “좋아요” 정보를 수정한다. 5. 웹 서비스에서 “좋아요” 버튼을 렌더링한다.
Alternate Flow	3.1. “좋아요” 상태인 경우 1. 웹 서비스가 프로젝트 “좋아요” 해제 요청을 보낸다. 2. 4 단계로 이동한다.
Activity Diagram	

Figure 19: Use Case 18 Activity Diagram

Table 63: Use Case 18

11. Architectural Design

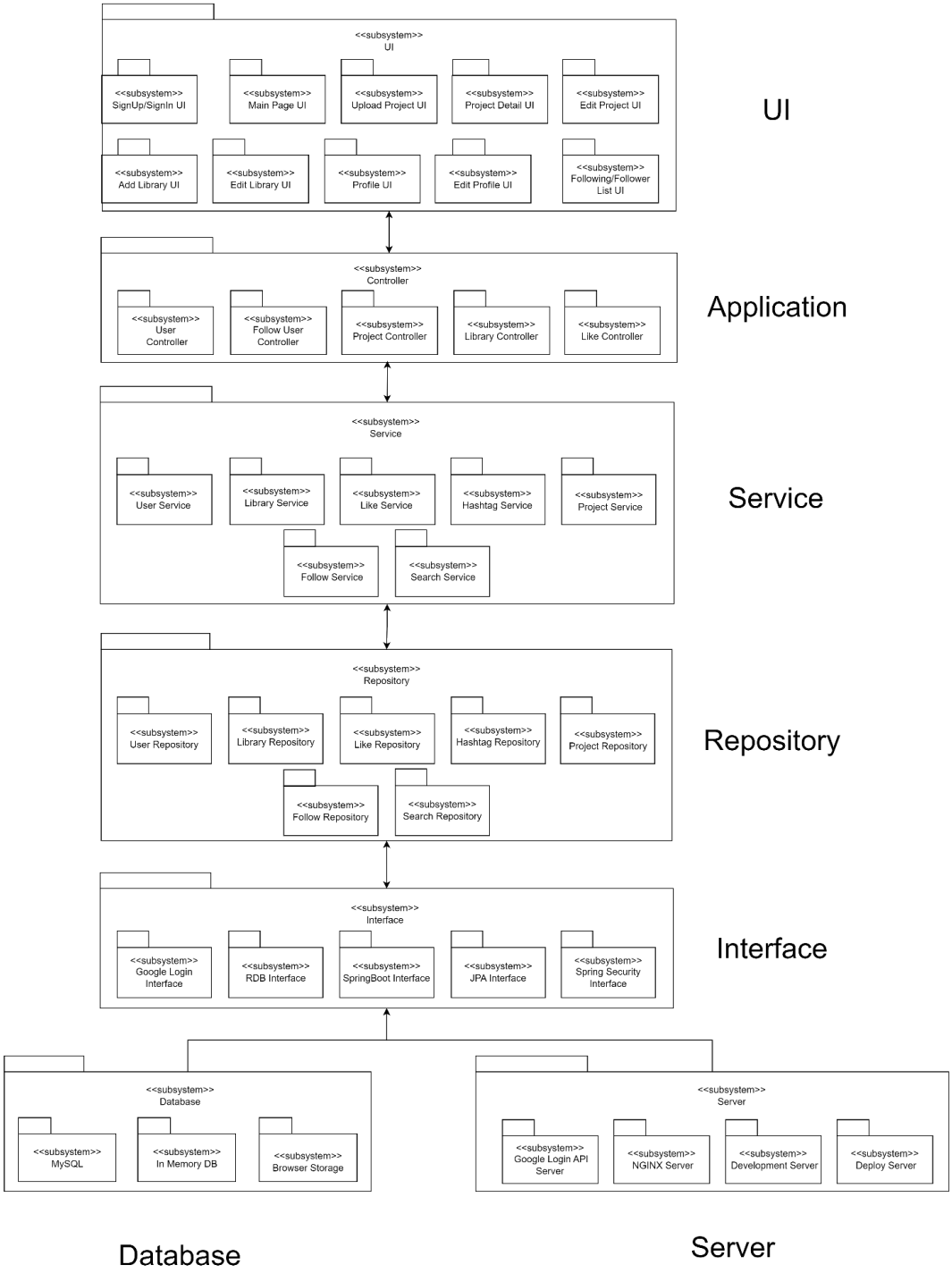


Figure 20: System Architecture Diagram

11.1 UI Layer

- 구성요소
    - SignUp/SignIn UI

        - 사용자가 시스템에 회원가입 및 로그인을 할 수 있다.
    - Main Page UI
        - 사용자가 시스템에 접속했을 때 가장 먼저 보게 되는 페이지.
        - 인기있는 라이브러리 목록, 내가 팔로우중인 사용자들의 프로젝트 목록을 확인할 수 있다.
    - Upload Project UI
        - 내 프로젝트를 업로드 할 수 있다.
    - Project Detail UI
        - 프로젝트의 상세정보를 확인할 수 있다.
    - Edit Project UI
        - 내가 업로드한 프로젝트를 수정할 수 있다.
    - Add Library UI
        - 프로젝트에 라이브러리를 등록할 수 있다.
    - Edit Library UI
        - 프로젝트에 등록된 라이브러리를 수정할 수 있다.
    - Profile UI
        - 사용자의 프로필을 확인할 수 있다.
    - Edit Profile UI
        - 내 프로필을 수정할 수 있다.
    - Following/Follower List UI
        - 내가 팔로우하는 사용자와 나를 팔로우하는 사용자의 목록을 확인할 수 있다.

11.2 Application Layer

- 구성요소
    - User Controller
        - 사용자가 시스템에 회원가입, 로그인, 로그아웃을 하도록 한다.
        - 시스템에 등록되어 있는 프로필을 조회, 수정한다.
    - Follow User Controller
        - 사용자가 다른 사용자를 팔로우, 언팔로우하게 한다.
    - Project Controller
        - 시스템에 프로젝트를 생성, 수정, 삭제하고 시스템에 등록된 프로젝트를 조회한다.
    - Library Controller

        - 시스템에 등록되어 있는 프로젝트에 라이브러리를 추가, 수정, 삭제하도록 한다.
    - Favorite Controller
        - 사용자가 시스템에 등록된 프로젝트에 좋아요를 등록, 해제한다.

11.3 Service Layer

- 구성요소
    - User Service
        - 사용자 관련 비즈니스 로직을 처리하는 서비스.
        - 회원 가입, 로그인, 프로필 관리 등 사용자 인증과 관련된 기능을 제공.
    - Library Service
        - 프로젝트 내에 있는 라이브러리들을 관리하는 서비스.
        - 라이브러리의 추가, 수정, 삭제 기능을 제공하며, 라이브러리의 상세 정보와 사용 방법을 볼 수 있도록 제공.
    - Project Service
        - 사용자가 자신의 프로젝트를 업로드하고 관리하는 서비스.
        - 프로젝트 업로드, 수정, 삭제 기능을 제공하며, 프로젝트에 포함된 라이브러리 정보들과 간단한 설명 제공.
    - Hashtag Service
        - 프로젝트나 라이브러리에 사용된 해시태그들을 관리하는 서비스.
        - 사용자가 프로젝트 검색할 때 해시태그를 포함해서 검색할 수 있어, 관련된 프로젝트를 찾는데 도움을 준다.
    - Follow Service
        - 사용자들 간의 팔로우 관계를 관리하는 서비스.
        - 사용자가 다른 사용자를 팔로우하거나, 다른 사용자가 자신을 팔로잉할 수 있으며, 팔로우한 프로젝트 또는 팔로잉한 사용자의 프로젝트를 볼 수 있다.
    - Search Service
        - 프로젝트와 관련된 검색 기능을 제공하는 서비스.
        - 사용자가 원하는 키워드로 프로젝트를 검색할 수 있다. 해시태그가 포함되어 프로젝트를 보다 정확하게 검색할 수 있다.

11.4 Interface Layer

- 구성요소

- Google Login Interface
  - 구글 로그인 인터페이스는 사용자가 구글 계정을 통해 인증 및 로그인하여 서비스를 이용할 수 있도록 제공한다.
- RDB Interface
  - RDB(관계형 데이터베이스) 인터페이스는 프로젝트에서 데이터베이스와의 상호작용을 담당한다.
  - 데이터베이스와의 연결, 데이터에 쿼리문을 처리하여 저장과 관리하는데 이용한다.
- Spring Boot Interface
  - Spring Boot 인터페이스는 프로젝트를 개발하고 실행하기 위한 환경을 구성하는 역할을 한다.
  - Spring Boot 프레임워크를 사용하여 프로젝트의 설정, 빌드, 실행 등을 관리한다.
- JPA Interface
  - JPA(Java Persistence API) 인터페이스는 객체와 데이터베이스 간의 매핑을 담당한다.
  - 객체 지향 프로그래밍에서 사용되는 클래스와 데이터베이스 테이블을 매핑하여 데이터의 저장, 조회, 변경 등을 수행.
  - SQL 쿼리를 직접 작성하지 않아도 데이터베이스 조작을 할 수 있게된다.
- Spring Security Interface
  - Spring Security 인터페이스는 프로젝트의 보안과 인증을 담당.
  - 사용자 인증과 접근 제어, 소셜 로그인 기능을 제공.
  - 프로젝트에서 사용자의 로그인, 회원 가입, 접근 제한 등을 처리하기 위해 사용한다.

11.5 Database Layer

- 구성요소
  - MySQL Database
    - MySQL 은 관계형 데이터베이스(RDB)를 담당하는 데이터베이스 관리 시스템(DBMS)이다.
    - 서비스에 필요한 정보를 저장, 수정하여 관리할 수 있는 기능을 제공.
  - In Memory Database

        - 유저의 세션을 저장해두는 데이터베이스이다.
        - 로그인한 유저라면 저장된 세션에서 정보를 가져와서 유저 정보 인증할 수 있다.
    - Browser Storage
        - 브라우저 저장소는 웹 브라우저 내에서 데이터를 클라이언트단에 저장하는 기능을 제공한다.
        - 클라이언트가 서버에서 받은 정보를 임시로 저장하는 용도로 쓰인다.

11.6 Server Layer

- 구성요소
    - Google Login API Server
        - Google Login API Server 는 사용자가 구글 계정을 통해 인증 및 로그인할 수 있도록 API 를 제공하는 서버.
        - 사용자의 인증 요청을 처리하고, 사용자 정보를 서비스에 반환한다.
        - 사용자의 인증과 로그인 회원가입 로직과 관련된다.
    - Development Server
        - Development Server 는 개발 환경에서 사용되는 서버로, 개발 과정에 있는 프로젝트를 빌드하고 배포하는 서버.
        - 개발 서버에서 테스트할 수 있다.
    - Deploy Server
        - Deploy Server 는 서비스가 실제로 운영하여 유저에게 제공할 수 있도록 배포하는 서버이다.
        - 실제 서비스를 운영 환경에서 실행하고 관리한다.

12. Design Analysis

12.1 User

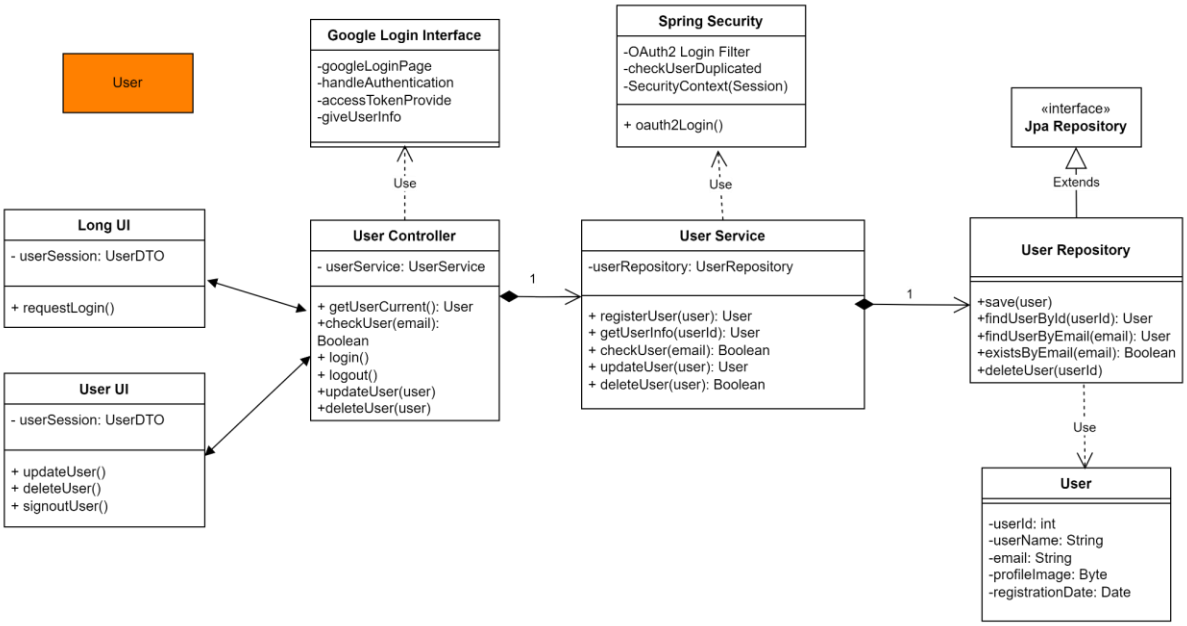


Figure 21: Design Class Diagram 01

- **Login UI Class**
  - 역할
    - Login UI Class 는 사용자 로그인 인터페이스(UI)를 담당하며, 로그인 화면을 구성하고, 구글 로그인을 진행할 수 있도록 사용자에게 화면을 제공한다.
  - 속성
    - userSession: 현재 로그인한 유저의 정보가 담긴 UserDTO 값을 가진다.
  - 메서드
    - requestLogin() : 사용자가 구글 로그인을 진행할 수 있도록 서버에게 요청한다.
- **User UI Class**
  - 역할
    - User UI Class 는 사용자 조회, 수정, 삭제의 인터페이스(UI)를 담당한다.
- **User Controller Class**
  - 역할

        - User Controller Class 는 사용자가 시스템에 회원가입, 로그인, 로그아웃 세션 관리를 하도록 하는 로직을 처리한다.
    - 속성
        - userService: UserService 객체를 사용하기 위한 필드이다.
    - 메서드
        - getUserCurrent() : 현재 세션에 있는 유저 정보를 가져와 클라이언트에 전달한다.
        - checkUser(email): 이미 가입한 유저인지 email 인자를 입력 받아서 검사할 수 있다.
        - login() : 구글 로그인을 진행한다.
        - logout() : 세션에 있는 유저 정보를 비운다.
        - updateUser(user): 유저의 정보를 수정할 때 쓴다.
        - deleteUser(user): 유저가 회원탈퇴 할 때 사용한다.
- **Google Login Interface**
    - 역할
        - Google Login 을 서비스에서 이용할 수 있도록 인터페이스를 제공한다.
    - 구성
        - googleLoginPage: 구글에서 제공하는 로그인 페이지.
        - handleAuthentication: 사용자의 구글 로그인 정보를 검증한다.
        - accessTokenProvider: 올바른 서버에서 정상적으로 유저가 구글 로그인에 성공한다면 토큰을 제공한다.
        - giveUserInfo: 발급받은 토큰을 이용해 구글 리소스 서버에서 유저 정보를 가져온다.
- **User Service Class**
    - 역할
        - User Controller 에서 필요한 비즈니스 로직 처리를 담당한다.
    - 속성
        - userRepository: 유저와 관련된 데이터베이스 로직을 사용할 때 사용한다.
    - 메서드
        - registerUser(user): 유저를 회원가입하는 로직에 쓰인다.
        - getUserInfo(userId): 유저의 정보를 가져올 때 쓴다.

        - checkUserEmail(email): 인자로 들어온 email 이 가입했던 email 인지 검사할 때 쓴다.
        - updateUser(user): 유저의 정보를 수정할 때 쓴다.
        - deleteUser(user): 유저 회원 탈퇴 할 때 쓴다.
- **Spring Security Class**
    - 역할
        - 웹 서비스에서 소셜 로그인과 사용자 인증, 접근 제한등의 보안 관련 기능을 담당한다.
    - 기능
        - OAuth2 Login Filter: 소셜 로그인을 진행해주는 필터이다.
        - checkUserDuplicated: 유저가 이미 가입한 유저인지 검사하는 메서드이다.
        - SecurityContext(Session): 로그인한 유저의 정보를 관리하는 Context 이다.
        - oauth2Login(): 소셜 로그인을 진행하는 필터 메서드이다.
- **User Repository Class**
    - 역할
        - 유저와 데이터베이스 관련 로직을 처리하는 클래스이다. Jpa Repository 를 상속하여 ORM 을 사용한다.
    - 메서드
        - save(user): 유저를 저장, 수정하는데 사용한다.
        - findUserById(userId): 유저를 ID 를 이용해 조회한다.
        - findUserByEmail(email): 유저의 email 을 이용해 조회한다.
        - existsByEmail(email): 유저가 가입한 email 인지 확인한다.
        - deleteUser(userId): 유저의 정보를 삭제하는데 사용한다.
- **User Class**
    - 역할
        - 데이터베이스에 저장되는 유저 테이블과 매핑되는 클래스이다.
    - 속성
        - userId: 유저의 PK.
        - userName: 유저의 이름.
        - email: 유저의 이메일.
        - profileImage: 유저의 프로필 이미지 URL 을 저장.

- registrationDate: 유저가 회원가입한 날짜를 저장.

12.2 Project

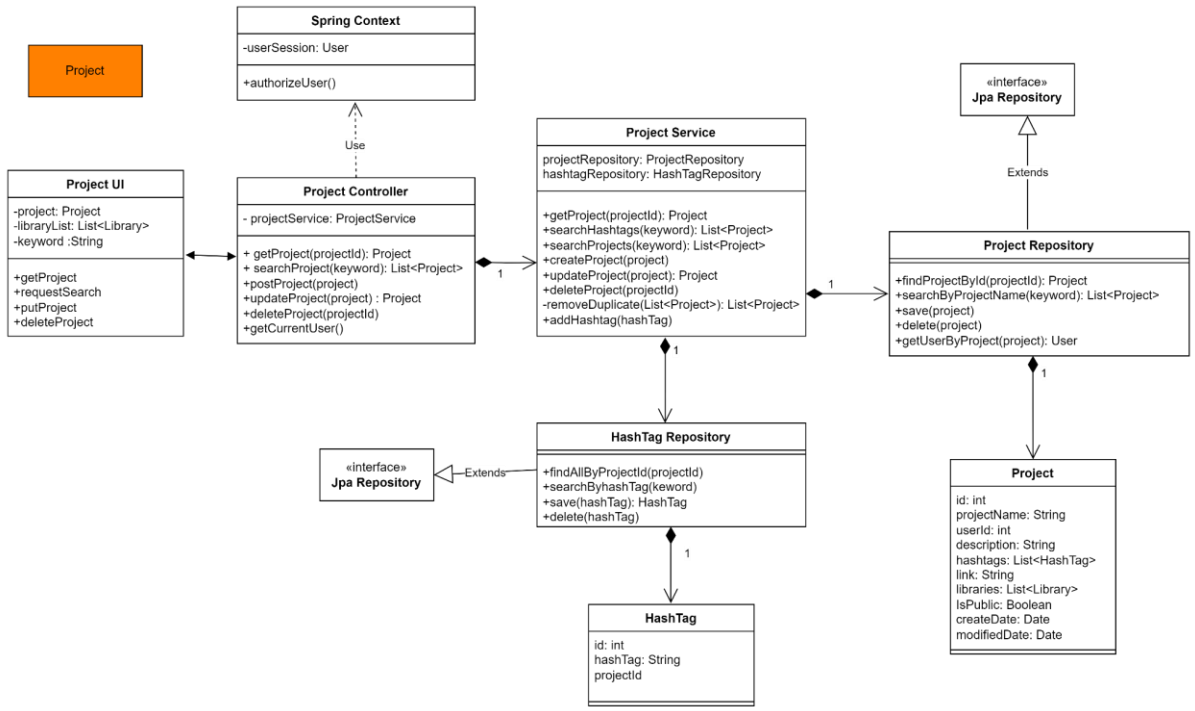


Figure 22: Design Class Diagram 02

- **Project UI**
  - 역할
    - Project UI Class 는 프로젝트를 이용할 수 있는 사용자 인터페이스(UI)를 담당하며, 사용자가 프로젝트를 조회, 관리할 수 있는 기능을 제공한다.
  - 구성
    - project: 클라이언트에서 Project 정보를 가진다. 그리고 이를 이용해 프로젝트 API 기능을 이용할 수 있다.
    - keyword: 키워드를 통한 프로젝트 검색이 가능하다.
    - getProject: 프로젝트를 조회할 수 있다.
    - requestSearch: 검색어로 프로젝트를 검색할 수 있다.
    - putProject: 프로젝트를 업데이트한다.
    - deleteProject: 프로젝트를 삭제한다.
- **Project Controller**

- 역할
  - Project Controller 는 클라이언트의 요청을 받아 프로젝트를 관리할 수 있는 기능을 제공한다.
- 속성
  - projectService: ProjectService: 프로젝트 관련 로직을 처리하는 클래스를 필드로 가진다.
- 메서드
  - getProject(projectId): 프로젝트 ID 를 이용해 조회한다.
  - searchProject(keyword): 키워드를 이용해 프로젝트 이름과 해시태그를 검색한다.
  - postProject(project): 프로젝트를 등록한다.
  - updateProject(project): 프로젝트를 수정한다.
  - deleteProject(projectId): 프로젝트를 삭제한다.
  - getCurrentUser(): 현재 세션에 저장된 유저를 가져온다.

- **Spring Context**
  - 역할
    - 세션에 로그인한 유저의 정보를 담는다. 이를 이용해서 유저를 가져와 Project 관련 기능을 수행할 때 유저가 권한이 있는지 검사한다.
  - 구성
    - userSession: 로그인한 유저의 정보를 가져올 수 있는 세션.
    - authorizeUser: 유저가 로직 관련 권한을 가지고 있는지 검사한다.
- **Project Service**
  - 역할
    - Project Service 는 사용자가 프로젝트에 관련 요청을 했을 때의 비즈니스 로직을 처리를 담당한다.
  - 속성
    - projectRepository: ProjectRepository : 프로젝트가 데이터베이스에 처리해야하는 요청을 사용할 때 필요하다.
    - hashtagRepository: 프로젝트의 해시태그들을 관리할 때 사용한다.
  - 메서드
    - getProject(projectId): 프로젝트 ID 를 이용해 조회한다.
    - searchHashTags(keyword): 검색어를 포함하는 해시태그를 가지는 Project 를 찾을 때 사용한다.

        - searchProjects(keyword): 검색어를 포함하는 프로젝트를 찾을 때 사용한다.
        - createProject(project): 프로젝트를 등록할 때 사용한다.
        - updateProject(project); 프로젝트의 해시태그나 라이브러리, 공개여부와 같은 내용을 수정할 때 사용한다.
        - deleteProject(projectId): 프로젝트를 삭제할 때 사용한다.
        - remoteDuplicate(List<Project>): 검색어를 포함하는 해시태그를 이용해 프로젝트를 가져왔을 때, 중복되는 프로젝트를 제거하고 프로젝트 목록으 t 로 반환한다.
        - addHashtag(hashTag): 프로젝트에 해시태그를 추가한다.
- **Project Repository**
    - 역할
        - 프로젝트와 관련된 데이터베이스 로직을 처리하는데 쓰인다. Jpa Repository 를 이용한 ORM 을 사용한다.
    - 메서드
        - findProjectById(projectId): 프로젝트를 조회한다.
        - searchByProjectName(keyword): 키워드를 포함하는 프로젝트를 가져온다.
        - save(project): 프로젝트를 등록, 수정할 때 사용한다.
        - delete(project): 프로젝트를 삭제할 때 사용한다.
        - getUserByProject(project): 프로젝트 소유자를 가져올 때 사용한다.
- **HashTag Repository**
    - 역할
        - 프로젝트의 해시태그들을 관리한다. Jpa Repository 를 이용한 ORM 을 사용한다.
    - 메서드
        - findAllByProjectIdTag(projectId): 프로젝트의 해시태그들을 가져올 때 사용한다.
        - searchByhashTag(keyword): 검색어를 포함하는 해시태그들을 가져온다.
        - save(hashTag): 프로젝트에 해시태그를 등록한다.
        - delete(hashTag): 프로젝트의 해시태그를 삭제한다.
- **Project**

- 역할
  - 데이터베이스에 저장되는 프로젝트 테이블과 매핑되는 클래스이다.
- 속성
  - id: 프로젝트의 고유 아이디.
  - projectName: 프로젝트 등록한 제목.
  - userId: 프로젝트 소유자의 고유 아이디.
  - description: 프로젝트 설명 저장.
  - hashtags: 프로젝트의 해시태그들.
  - link: 프로젝트의 링크.
  - libraries: 프로젝트 내에 등록한 라이브러리들.
  - isPublic: 프로젝트의 공개 여부를 결정한다.
  - createDate: 프로젝트 등록 날짜.
  - modifiedDate: 프로젝트 수정 날짜.

- **HashTag**
  - 역할
    - 데이터베이스에 저장되는 해시태그 테이블과 매핑되는 클래스이다.
  - 속성
    - id: 해시태그 고유 ID.
    - hashTag: 해시태그 내용.
    - projectId: 해시태그를 소유한 프로젝트의 고유 ID.

12.3 Library

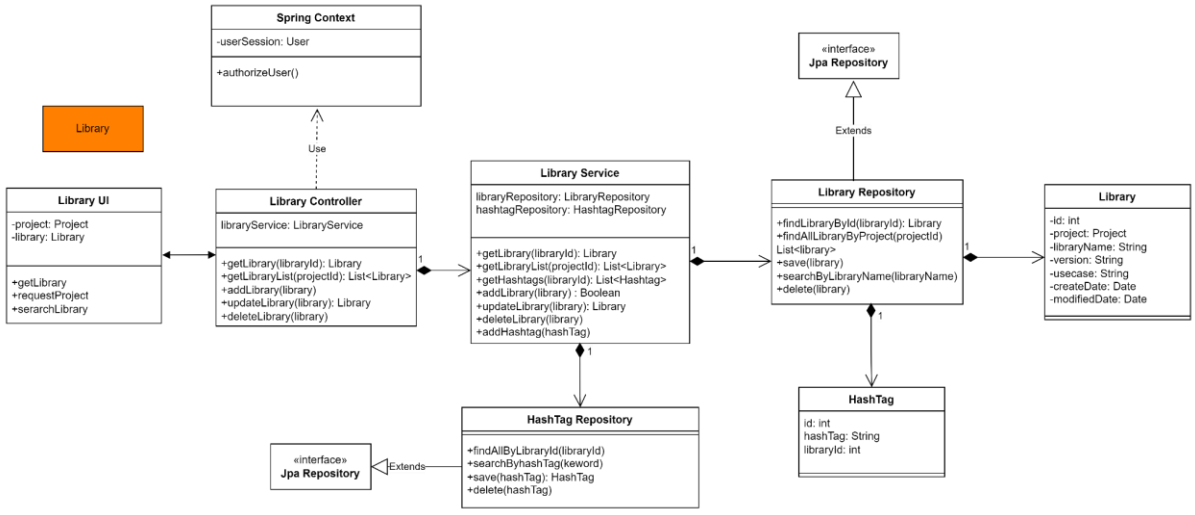


Figure 23: Design Class Diagram 03

- **Library UI**

    - 역할
        - Library UI Class 는 프로젝트 내에서 라이브러리를 관리할 수 있는 사용자 인터페이스(UI)를 담당하며, 사용자가 라이브러리를 조회, 관리할 수 있는 기능을 제공한다.
    - 구성
        - project: 현재 라이브러리를 소유한 프로젝트의 정보를 저장한다.
        - library: 현재 라이브러리의 정보를 가진다.
        - getLibrary: 라이브러리 정보를 가져온다.
        - requestProject: 라이브러리를 소유한 프로젝트 정보를 가져온다.
        - searchLibrary: 검색어를 이용해 라이브러리를 검색한다.
- **Library Controller**
    - 역할
        - 라이브러리 관련 요청을 받아서, 해당 요청을 처리를 담당하는 기능을 제공한다.
    - 속성
        - getLibrary(libraryId): 특정 라이브러리를 조회한다.
        - getLibraryList(projectId): 프로젝트가 소유한 라이브러리 목록을 조회한다.
        - addLibrary(library): 라이브러리를 프로젝트에 추가한다.
        - updateLibrary(library): 프로젝트 내에 있는 라이브러리의 내용을 수정한다.
        - deleteLibrary(library): 프로젝트 내에 있는 라이브러리를 삭제한다.
- **Spring Context**
    - 역할
        - 세션에 로그인한 유저의 정보를 담는다. 이를 이용해서 유저를 가져와 라이브러리와 관련 기능을 수행할 때 유저가 권한이 있는지 검사한다.
    - 구성
        - userSession: 로그인한 유저의 정보를 저장해둔 세션이다. 요청한 유저의 정보를 가져올 때 필요하다.
        - authorizeUser: 유저가 로직 관련 권한을 가지고 있는지 검사한다.
- **Library Service**
    - 역할
        - 라이브러리 관련 요청의 비즈니스 로직을 처리하는 서비스 계층이다.

- 속성
  - libraryRepository: 라이브러리가 데이터베이스에 처리해야하는 요청을 사용할 때 필요하다.
  - hashtagRepository: 프로젝트의 해시태그들을 관리할 때 사용한다
- 메서드
  - getLibrary(libraryId): 라이브러리를 조회한다.
  - getLibraryList(projectId): 프로젝트가 소유한 라이브러리 목록을 가져온다
  - getHashTags(libraryId): 라이브러리가 소유한 해시태그들을 가져온다
  - addLibrary(library): 프로젝트에 라이브러리를 추가한다
  - updateLibrary(library): 라이브러리 정보를 수정한다
  - deleteLibrary(library): 라이브러리 정보를 삭제한다

- **Library Repository**
  - 역할
    - 라이브러리 관련 데이터베이스 로직을 처리하는 클래스이다. Jpa Repository 를 이용한 ORM 을 사용한다.
  - 메서드
    - findLibraryById(libraryId): 라이브러리 ID 와 일치하는 라이브러리를 데이터베이스에서 조회한다
    - findAllLibraryByProject(projectId): 프로젝트가 소유하는 라이브러리들을 가져온다
    - save(library): 라이브러리를 수정, 저장한다
    - searchByLibraryName(libraryName): 라이브러리 이름 중 검색어를 포함하는 것들을 검색한다
    - delete(library): 라이브러리를 삭제한다
- **HashTag Repository**
  - 역할
    - 라이브러리의 해시태그들을 관리한다. Jpa Repository 를 이용한 ORM 을 사용한다.
  - 메서드
    - findAllByLibraryId(libraryId): 라이브러리의 해시태그들을 가져올 때 사용한다

        - searchByhashTag(keyword): 검색어를 포함하는 해시태그들을 가져온다
        - save(hashTag): 프로젝트에 해시태그를 등록한다
        - delete(hashTag): 프로젝트의 해시태그를 삭제한다
- **Library**
    - 역할
        - 데이터베이스에 저장되는 라이브러리 테이블과 매핑되는 클래스이다
    - 속성
        - id: 라이브러리의 고유 ID.
        - project: 라이브러리를 소유한 프로젝트.
        - libraryName: 라이브러리 이름.
        - version: 라이브러리 버전.
        - usecase: 라이브러리 관련 설명 저장.
        - createDate: 라이브러리 정보 생성 시간.
        - modifiedDate: 라이브러리 정보 수정 시간.
- **HashTag**
    - 역할
        - 데이터베이스에 저장되는 해시태그 테이블과 매핑되는 클래스이다.
    - 속성
        - id: 해시태그 고유 ID.
        - hashTag: 해시태그 내용.
        - librarytId: 해시태그를 소유한 라이브러리의 고유 ID.

12.4 Favorite

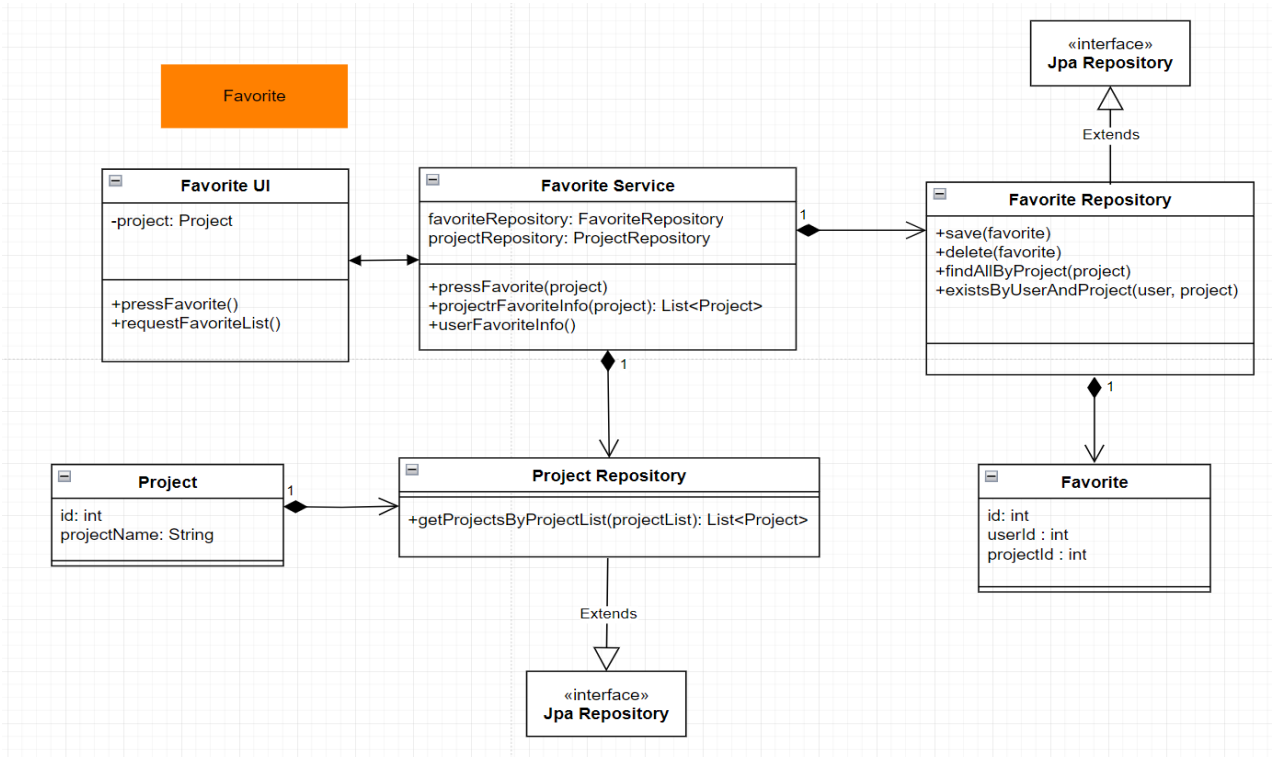


Figure 24: Design Class Diagram 04

- **Favorite UI**
  - 역할
    - Favorite UI Class 는 관심있는, 맘에 드는 프로젝트를 좋아요 표시할 수 있는 인터페이스(UI)를 담당한다.
  - 구성
    - pressFavorite: 좋아요 표시, 좋아요 취소. 이미 좋아요가 눌려진 상태라면 좋아요 취소 요청을 날린다.
    - requestFavoriteList: 좋아요 누른 프로젝트 리스트를 가져온다.
- **Favorite Controller**
  - 역할
    - 좋아요 요청을 받아서, 해당 요청을 처리를 담당하는 기능을 제공한다.
  - 속성
    - favoriteService: 좋아요 관련 비즈니스 로직이 필요할 때 호출해서 사용한다.
  - 메서드
    - pressFavorite(project): 프로젝트에 좋아요 표시.
    - cancelFavorite(project): 프로젝트에 좋아요한 표시 취소.

        - getFavoriteProjects(): 좋아요 누른 프로젝트들을 가져온다.
- **Favorite Service**
    - 역할
        - 좋아요 관련 요청의 비즈니스 로직을 처리하는 서비스 계층이다.
    - 속성
        - favoriteRepository: 좋아요 관련 데이터베이스 처리가 필요할 때 사용한다.
        - projectRepository: 프로젝트 관련한 데이터를 데이터베이스에서 가져와야할 때 쓴다.
    - 메서드
        - pressFavorite(project): 프로젝트에 좋아요 표시.
        - cancelFavorite(project): 프로젝트에 좋아요한 표시 취소.
        - getFavoriteProjects(): 좋아요 누른 프로젝트들을 가져온다.
- **Project Repository**
    - 역할
        - 좋아요 누른 프로젝트 정보를 데이터베이스에서 가져올 때 사용한다. Jpa Repository 를 이용한 ORM 을 사용한다.
    - 메서드
        - findFavoriteProjectsByUserId(userId): 유저가 좋아요 누른 프로젝트들을 가져올 때 사용한다.
- **Favorite Repository**
    - 역할
        - 좋아요 누른 정보를 데이터베이스에 저장, 삭제할 때 필요하다. Jpa Repository 를 이용한 ORM 을 사용한다.
    - 메서드
        - save(favorite): 좋아요 저장.
        - delete(favorite): 좋아요 취소 (삭제).
- **Favorite**
    - 역할
        - 데이터베이스에 저장되는 좋아요 테이블과 매핑되는 클래스이다.
    - 속성
        - id: 좋아요의 고유 ID.
        - userId: 좋아요 누른 유저의 아이디.

        - projectId: 좋아요 누른 프로젝트 아이디.
- **Project**
    - 역할
        - 좋아요 누른 프로젝트들의 정보들이다.
    - 속성
        - id: 좋아요 누른 프로젝트 고유 ID.
        - projectName: 좋아요 누른 프로젝트 이름.

12.5 Follow

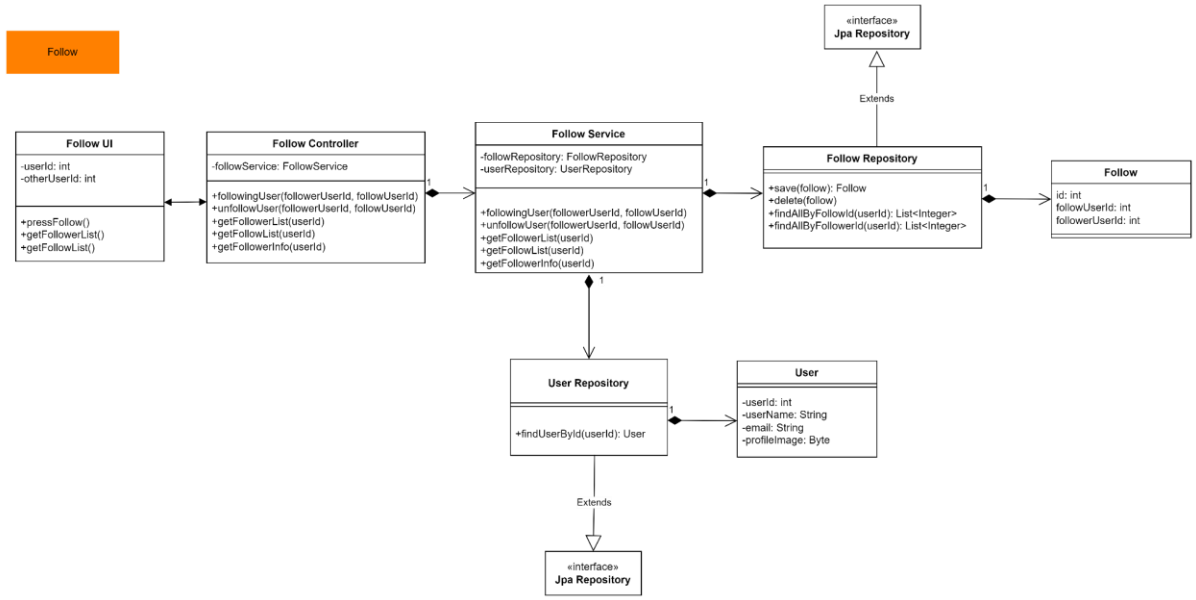


Figure 25: Design Class Diagram 05

- **Follow UI**
    - 역할
        - Follow UI Class 는 관심있는, 맘에 드는 유저를 팔로우하거나, 자신의 팔로워 목록을 볼 수 있는 사용자 인터페이스(UI)를 담당한다.
    - 구성
        - userId: 자신의 ID.
        - otherUserId: 팔로우한 유저의 ID.
        - pressFollow: 팔로우 기능 요청.
        - getFollowerList: 팔로워 목록을 가져온다.
        - getFollowList: 자신을 팔로잉하는 유저들의 목록을 가져온다.
- **Follow Controller**
    - 역할

        - 팔로우, 팔로워, 팔로잉 목록 요청을 받아서, 해당 요청을 처리를 담당하는 기능을 제공한다.
    - 속성
        - followService: FollowService 는 팔로우 관련 비즈니스 로직을 처리하기 위해 사용한다.
    - 메서드
        - followingUser(followerUserId, followUserId): 사용자를 팔로우한다.
        - unfollowUser(followerUserId, followUserId): 팔로우했던 사용자를 취소한다.
        - getFollowerList(userId): 자신의 팔로워들의 목록을 가져온다.
        - getFollowList(userId): 팔로우하는 유저들의 목록을 가져온다.
        - getFollowerInfo(userId): 팔로워의 정보를 가져온다.
- **Follow Service**
    - 역할
        - 팔로우 관련 비즈니스 로직 처리를 담당하는 클래스이다.
    - 속성
        - followRepository: FollowRepository 는 팔로우 관련 데이터베이스 처리를 하기 위해 사용한다.
        - userRepository: UserRepositry 는 팔로우, 팔로잉 유저 관련 데이터베이스 요청을 처리하기 위해 사용한다.
    - 메서드
        - followingUser(followerUserId, followUserId): 사용자를 팔로우한다.
        - unfollowUser(followerUserId, followUserId): 팔로우했던 사용자를 취소한다.
        - getFollowerList(userId): 자신의 팔로워들의 목록을 가져온다.
        - getFollowList(userId): 팔로우하는 유저들의 목록을 가져온다.
        - getFollowerInfo(userId): 팔로워의 정보를 가져온다.
- **Follow Repository**
    - 역할
        - 팔로우 관련 데이터베이스 로직을 처리한다. Jpa Repository 를 이용한 ORM 을 사용한다.
    - 메서드

        - save(follow): 팔로우 정보를 저장한다.
        - delete(follow): 팔로우 정보를 삭제한다.
        - findAllByFollowId(userId): 팔로잉 목록을 조회한다.
        - findAllByFollowerid(userId): 팔로워 목록을 조회한다.
- **User Repository**
    - 역할
        - 유저의 정보를 데이터베이스에서 가져올 때 필요하다. Jpa Repository 를 이용한 ORM 을 사용한다.
    - 메서드
        - findUserById(userId): 유저 ID 로 유저를 조회한다.
- **Follow**
    - 역할
        - 데이터베이스에 저장되는 좋아요 테이블과 매핑되는 클래스이다.
    - 속성
        - id: 팔로우 고유 ID.
        - followUserId: 자신이 팔로잉하는 유저들의 ID 를 저장한다.
        - followerUserId: 자신의 팔로워 유저들의 ID 를 저장한다.
- **User**
    - 역할
        - 팔로워 유저나 팔로잉 유저들과 데이터베이스가 매핑되는 클래스이다.
    - 속성
        - userId: 팔로워 또는 팔로잉 유저의 고유 ID.
        - userName: 팔로워 또는 팔로잉 유저의 이름.
        - email: 팔로워 또는 팔로잉 유저의 이메일.
        - profileImage: 팔로워 또는 팔로잉 유저의 프로필 이미지 URL 을 저장.

13. Use case Realization

13.1 로그인 / 회원가입 / 로그아웃

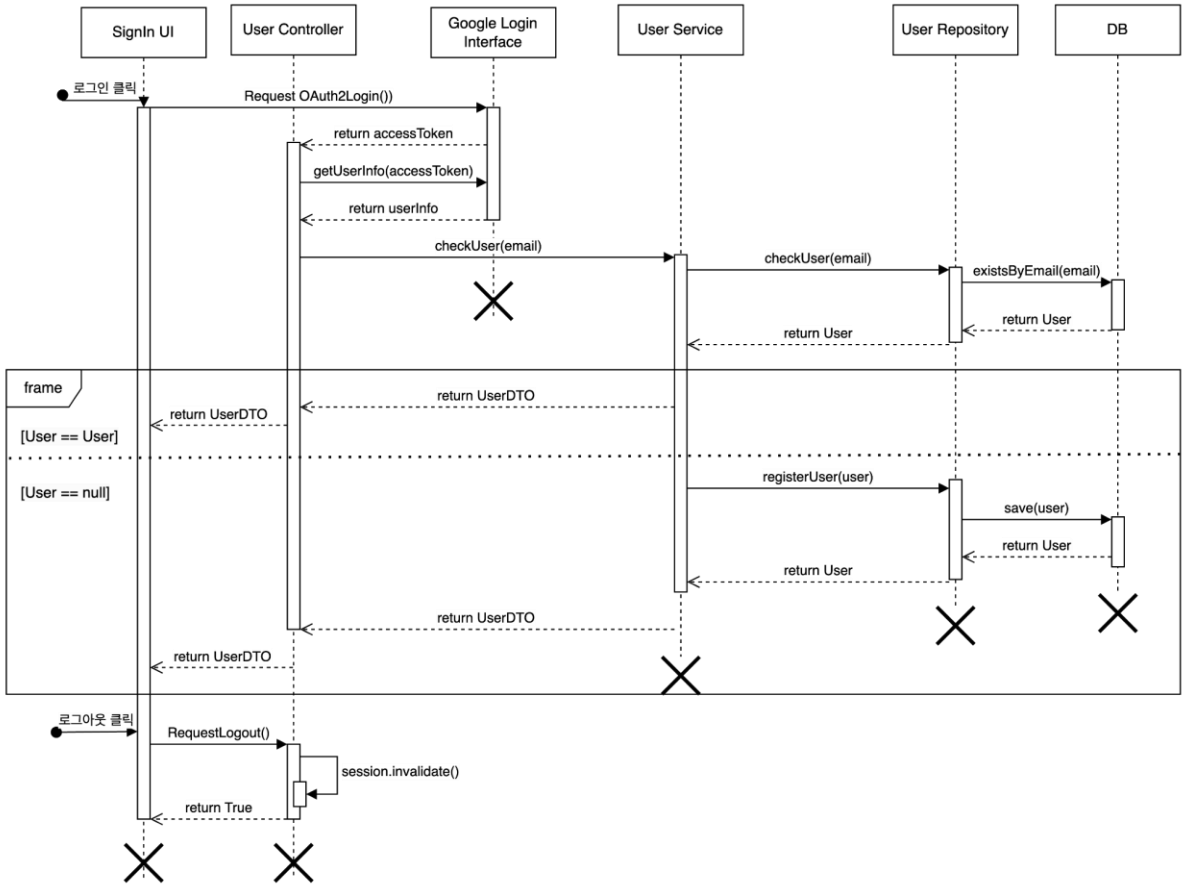


Figure 26: Sequence Diagram 01

사용자가 로그인 버튼을 클릭하면, 클라이언트는 Google Login Interface 에게 OAuth2Login() 요청을 보내 인증을 요청하고 User Controller 가 액세스 토큰을 받아 다시 Google Login Interface 에게 사용자 정보를 요청한다. 받아온 정보를 토대로 User Service 가 User Repository 의 existsByEmail()을 사용하여 데이터베이스에 해당 이메일을 갖고 있는 사용자가 존재하는지 확인하고, 없으면 User Repository 의 save()을 사용하여 새로운 사용자를 등록한다. 사용자가 등록되거나 확인되면, 사용자 데이터가 DTO 로 변환되어 클라이언트에게 반환 된다. 로그아웃 요청 시, UserController 는 세션에서 해당 유저의 정보를 지우고 로그아웃이 성공했음을 나타내는 true 값을 반환한다.

13.2 프로젝트 검색 / 조회

13.2.1 프로젝트 검색

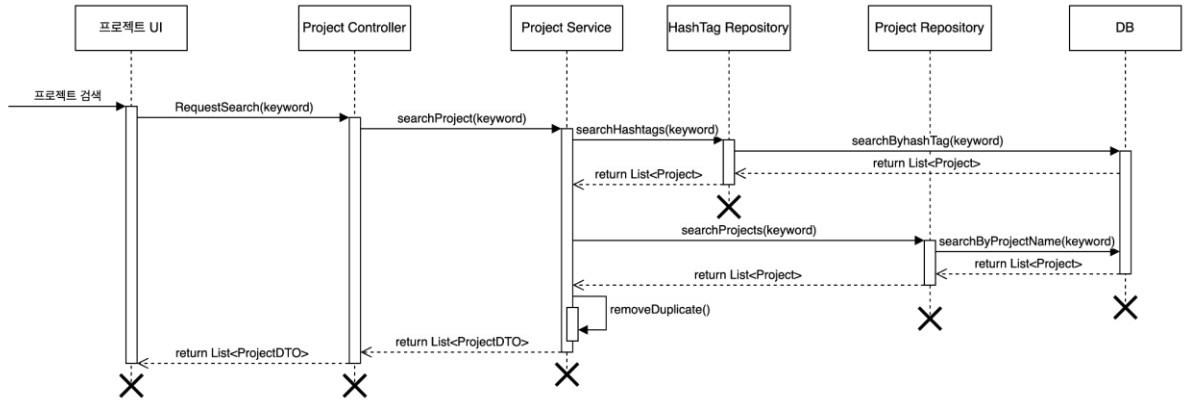


Figure 27: Sequence Diagram 02

사용자는 프로젝트 UI 를 통해 특정 키워드로 프로젝트를 검색하는 요청을 한다. 이 요청은 Project Controller 로 전달되고, Project Service 는 두 개의 Repository 를 통해 검색을 수행합니다: 해시태그를 통한 검색과 프로젝트 이름을 통한 검색이다. 이 결과는 중복을 제거하는 처리 과정을 거쳐, 최종적으로 사용자에게 반환되는 프로젝트 DTO 로 통합되어 클라이언트에게 반환 된다.

13.2.2 프로젝트 조회

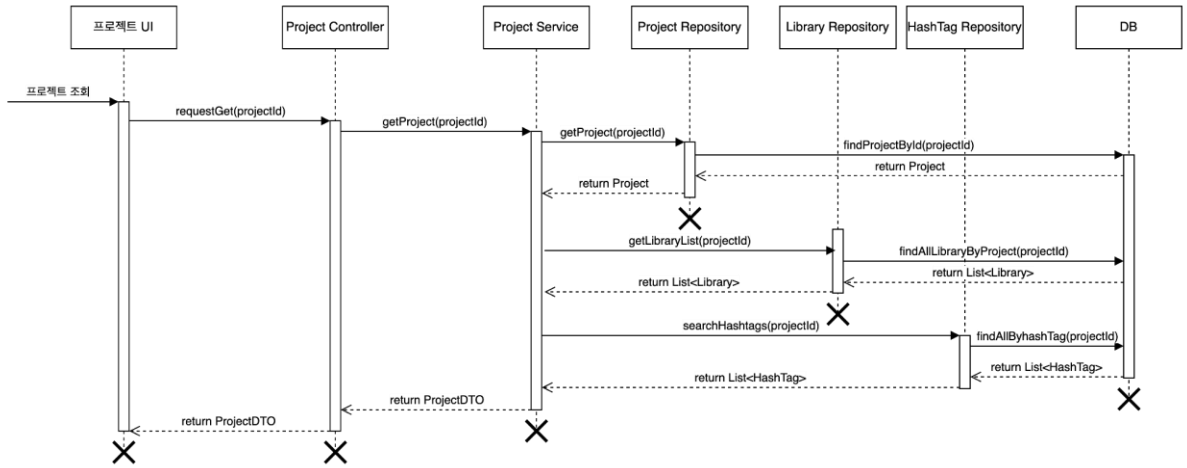


Figure 28: Sequence Diagram 03

사용자가 프로젝트 UI 를 통해 특정 프로젝트의 상세 정보를 요청하면, 이 요청은 Project Controller 로 전송된다. Project Controller 는 Project Service 에 projectId 를 전달하여 정보를 조회한다. Project Service 는 Project Repository, Library Repository 및 Hashtags Repository 로부터 필요한 데이터를 가져온다. Project Repository 는 프로젝트 정보를, Library Repository 는 프로젝트에 속한 라이브러리 목록을, Hashtags Repository 는 프로젝트에 연관된 해시태그 목록을 제공한다. 이 모든 정보는 프로젝트 DTO 로 통합되어 클라이언트에게 반환된다.

13.3 프로젝트 생성 / 수정 / 삭제

13.3.1 프로젝트 생성

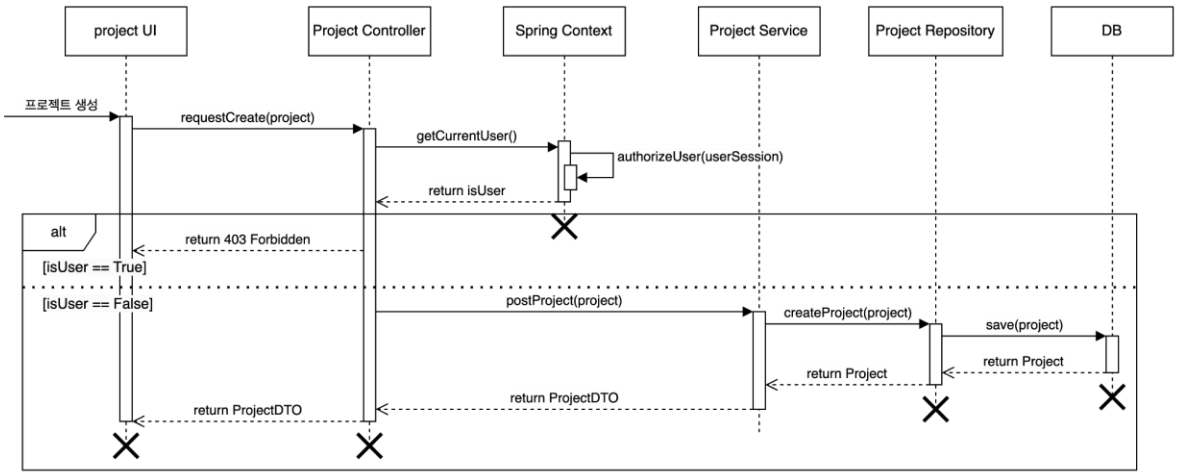


Figure 29: Sequence Diagram 04

사용자가 프로젝트 생성을 요청하면, Project Controller 는 사용자의 세션 정보를 검증하기 위해 Spring Context 에 있는 사용자 인증 메소드를 호출한다. 인증이 성공하면 Project Service 는 Project Repository 를 통해 프로젝트 정보를 데이터베이스에 저장하고, 저장된 프로젝트 정보를 DTO 형태로 변환하여 UI 로 다시 전달한다. 사용자가 유효하지 않거나 권한이 없는 경우, 시스템은 403 Forbidden 오류를 반환하여 사용자가 요청을 수행할 수 없음을 알린다.

13.3.2 프로젝트 수정

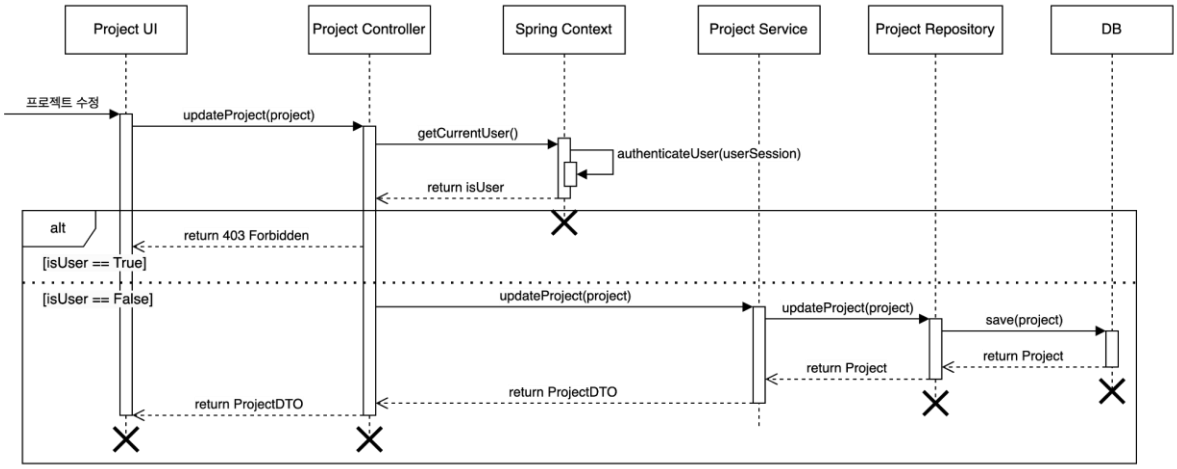


Figure 30: Sequence Diagram 05

사용자가 프로젝트 UI 를 통해 업데이트를 요청하면 Project Controller 는 현재 사용자의 유효성을 확인하기 위해 Spring Context 에 접근한다. 사용자가 유효하다면, Project Service 는 사용자 세션을 인증하고, 해당 프로젝트를 업데이트하는 과정을 진행한다. 이 과정은 Project Repository 에 프로젝트 정보를 저장하고 결과를 Project DTO 로 변환하여 프로젝트 UI 에 반환하는 것을 포함한다. 만약 사용자가 유효하지 않으면, 403 Forbidden 오류가 반환되어 접근이 거부된다.

13.3.3 프로젝트 삭제

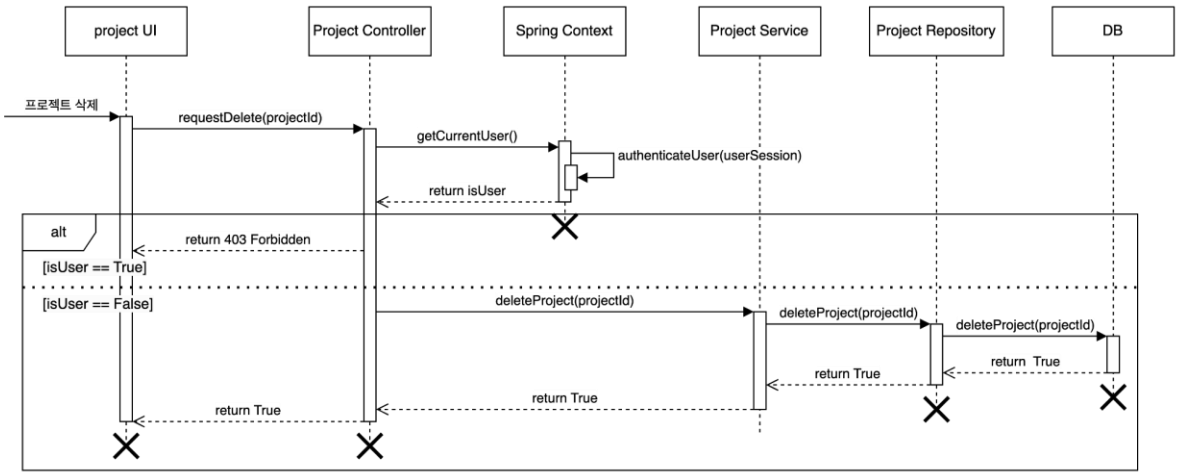


Figure 31: Sequence Diagram 06

사용자가 삭제 요청을 하면, Project Controller 는 Spring Context 를 통해 현재 사용자가 유효한지 확인한다. 사용자가 유효하지 않으면 403 Forbidden 오류를 반환하고, 유효한 사용자인 경우 Project Service 는 사용자의 세션을 인증한다. 인증 후, Project Service 는 Project Repository 에 deleteProject() 메소드를 호출하여 프로젝트를 삭제하고, 성공 여부를 나타내는 boolean 값을 반환한다. 삭제가 성공적으로 완료되면, 최종적으로 클라이언트에게 true 를 반환한다.

13.4 라이브러리 생성 / 수정 / 삭제

13.4.1 라이브러리 생성

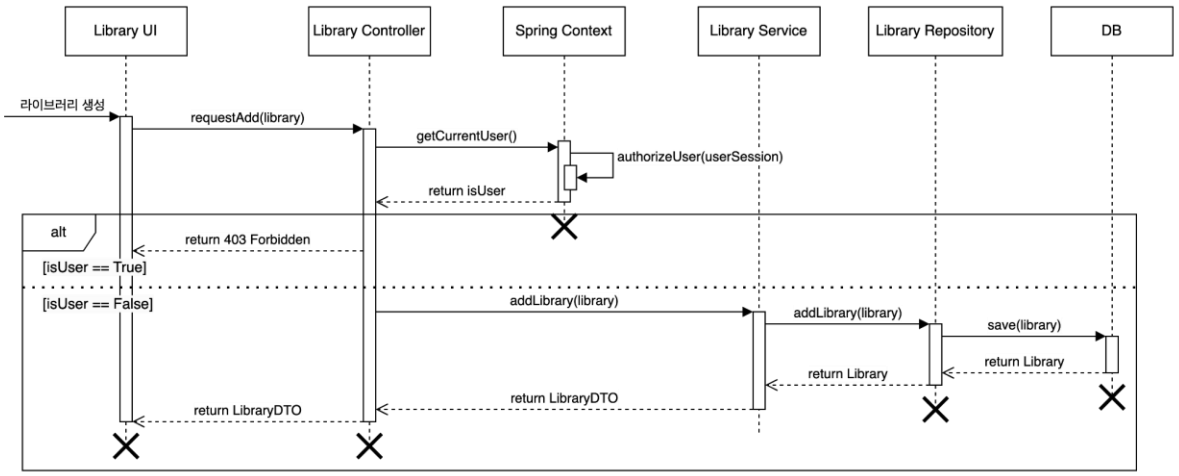


Figure 32: Sequence Diagram 07

사용자가 라이브러리 UI 를 통해 추가를 요청하면, Library Controller 가 Spring Context 를 통해 현재 로그인한 사용자가 유효한지 확인한다. 유효한 사용자라면, Library Service 이 사용자의 세션을 인증하고 Library Repository 를 통해 새 라이브러리 항목을 데이터베이스에 추가한다. 추가된 라이브러리 항목은 DTO 형태로 변환되어 UI 로 반환되며, 이를 통해 사용자는 새로운

라이브러리 항목이 성공적으로 추가되었음을 확인할 수 있다. 만약 사용자가 유효하지 않을 경우, 시스템은 403 Forbidden 오류를 반환하여 접근을 거부한다.

13.4.2 라이브러리 수정

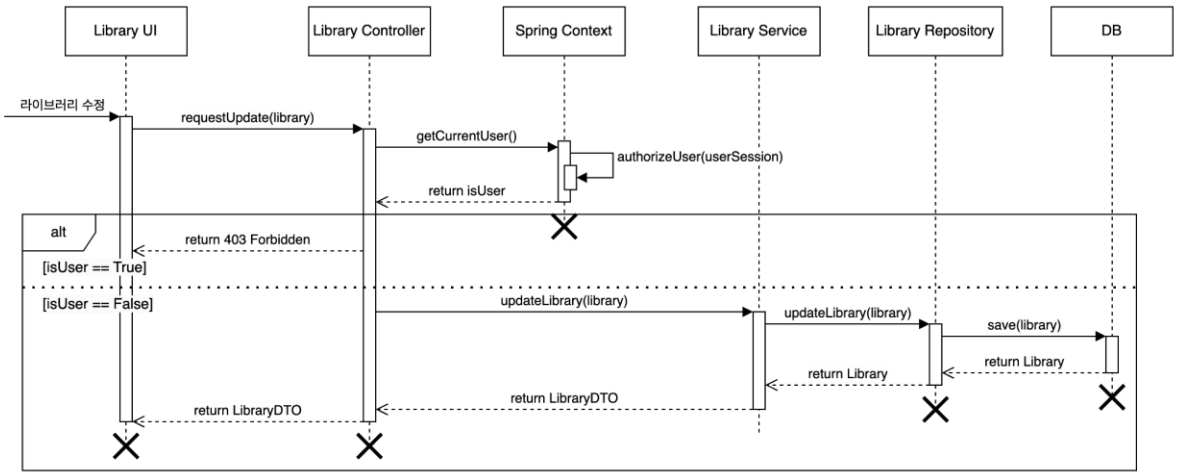


Figure 33: Sequence Diagram 08

사용자가 라이브러리 UI 에서 업데이트를 요청하면, Library Controller 는 Spring Context 를 통해 사용자의 인증 상태를 확인한다. 인증된 사용자는 Library Service 를 통해 라이브러리 업데이트 요청을 진행할 수 있다. 이 서비스는 Library Repository 에 업데이트 요청을 보내 데이터베이스에 저장하고, 저장된 결과를 라이브러리 DTO 로 변환하여 클라이언트에게 반환한다. 인증되지 않은 사용자가 요청을 할 경우, 접근이 거부되고 403 Forbidden 응답을 받게 된다.

13.4.3 라이브러리 삭제

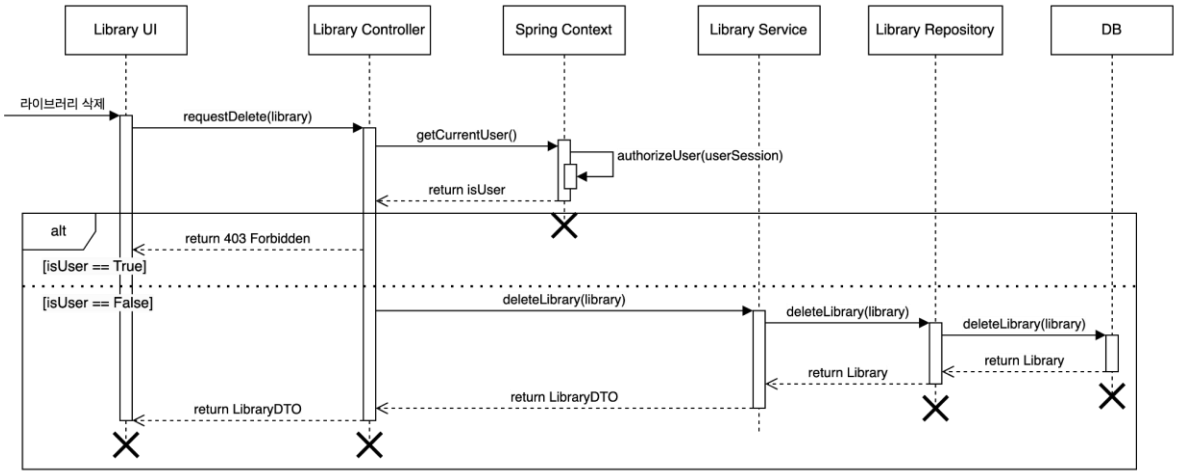


Figure 34: Sequence Diagram 09

사용자가 라이브러리 UI 를 통해 삭제 요청을 하면, Library Controller 는 사용자가 시스템에 유효한지 확인하기 위해 Spring Context 의 인증 기능을 호출한다. 사용자가 인증되면, Library Service 는 삭제 권한을 검사하고, Library Repository 를 통해 해당 항목을 데이터베이스에서 삭제한다. 삭제 작업이 성공하면, 결과가 라이브러리 DTO 로 변환되어 클라이언트에게 반환된다. 사용자 인증에 실패할 경우, 403 Forbidden 오류가 반환된다.

13.5 좋아요 / 좋아요 취소

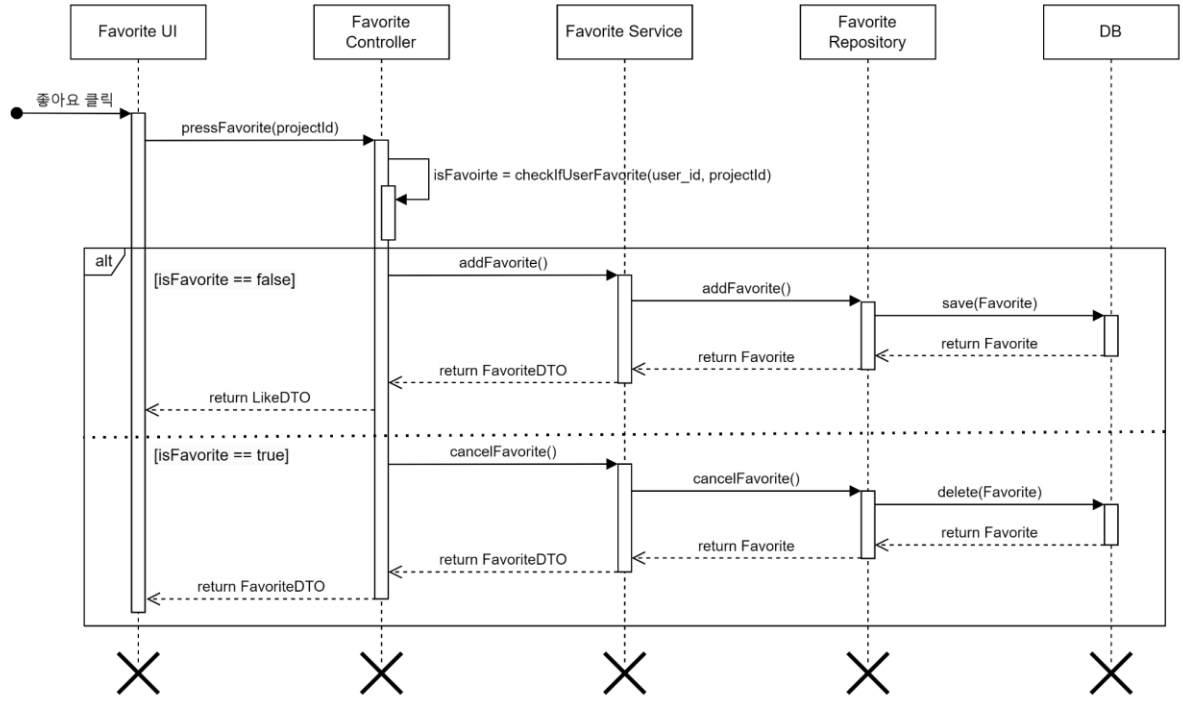


Figure 35: Sequence Diagram 10

등록된 사용자는 마음에 드는 프로젝트에 대하여 “좋아요” 등록을 할 수 있고, 이미 등록되어 있는 경우 “좋아요” 해제하도록 작동해야한다.

Favorite UI 가 제공한 버튼을 클릭하면 Favorite Controller 는 이 사용자가 이미 좋아요한 상태인지를 판단한다. checkIfUserFavorite() 결과에 따라 Favorite Controller 는 Favorite Service 에 등록 및 삭제 요청을 보낸다. Favorite Service 는 Favorite 정보를 등록 및 삭제하기 위해 Favorite Repository 를 이용하여 데이터베이스 동작을 수행한다. 데이터베이스 동작이 수행되면, Favorite UI 및 Favorite Controller 에 요청이 처리됐음을 알리기 위하여 Favorite 정보를 반환한다.

13.6 관심 프로젝트 목록 조회

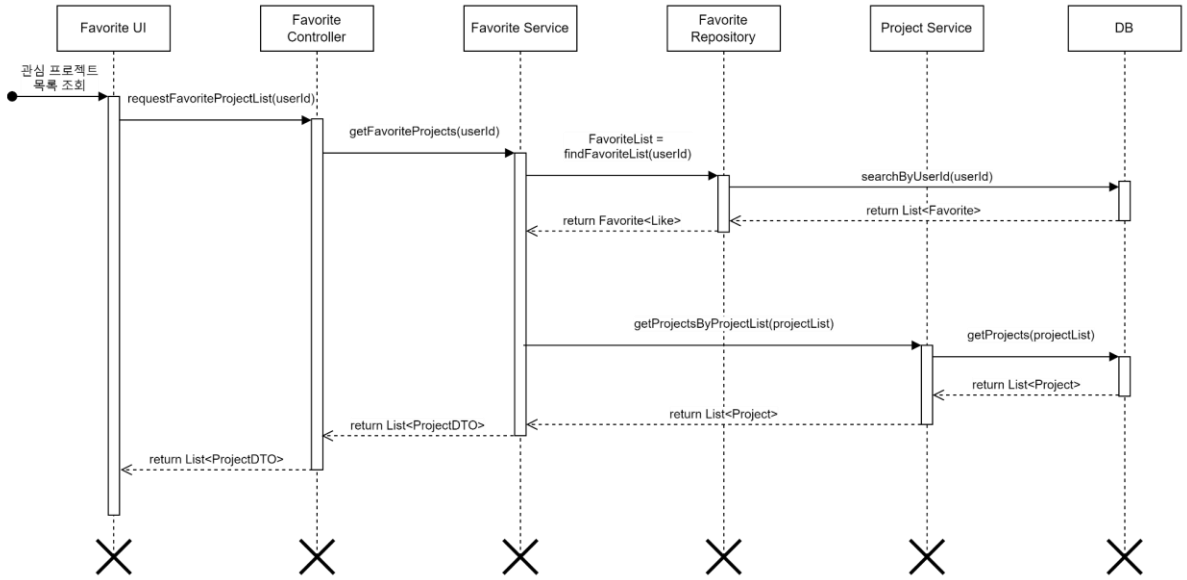


Figure 36: Sequence Diagram 11

등록된 사용자는 자신이 “좋아요”한 모든 프로젝트 목록을 제공 받을 수 있어야한다.
Like UI 가 프로젝트 목록 조회를 위한 페이지 요청을 받으면 Like Controller 는 Like Service 에 좋아요한 프로젝트 리스트를 요청한다. Like Service 는 자신이 등록한 좋아요 정보 리스트를 얻기 위해 Like Repository 를 이용하여 데이터베이스 동작을 수행하고 좋아요 정보 리스트를 반환받는다. 좋아요 정보 리스트를 반환받은 Like Service 는 Project Service 를 이용하여 프로젝트 리스트를 반환받는다. Like Service 는 Like UI 와 Like Controller 에게 프로젝트 리스트를 전달한다.

13.7 팔로우 / 언팔로우

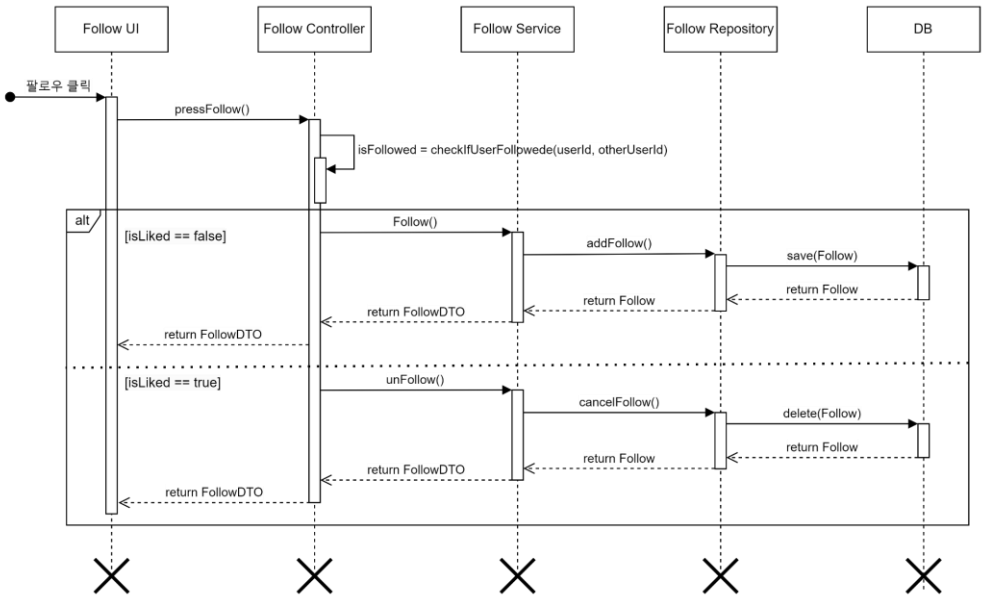


Figure 37: Sequence Diagram 12

동작 과정이 ‘4.6 좋아요 / 좋아요 취소’와 매우 유사하다. Follow Controller 에서 사용자가 이미 팔로우 했는지 확인하여 팔로우 상태에 따라 Follow Service 에 팔로우 요청 혹은 팔로우 해제 요청을 처리한다. Follow Service 가 Follow Repository 를 이용하여 데이터베이스에 접근하고 팔로우 정보를 처리한다.

13.8 프로필 조회 및 수정

13.8.1 프로필 조회

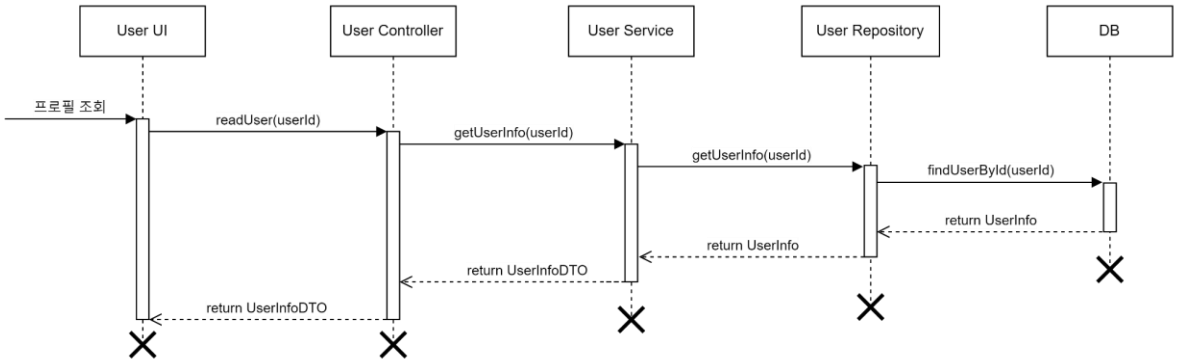


Figure 38: Sequence Diagram 13

사용자가 프로필 조회를 요청하면, User Controller 는 User Service 에 유저 정보를 요청한다. User Repository 를 통해 유저 정보를 얻고, 저장된 프로젝트 정보를 DTO 형태로 변환하여 UI 로 다시 전달한다. 사용자 정보는 모두에게 공개되어 별도의 검증을 하지 않는다.

13.8.2 프로필 수정

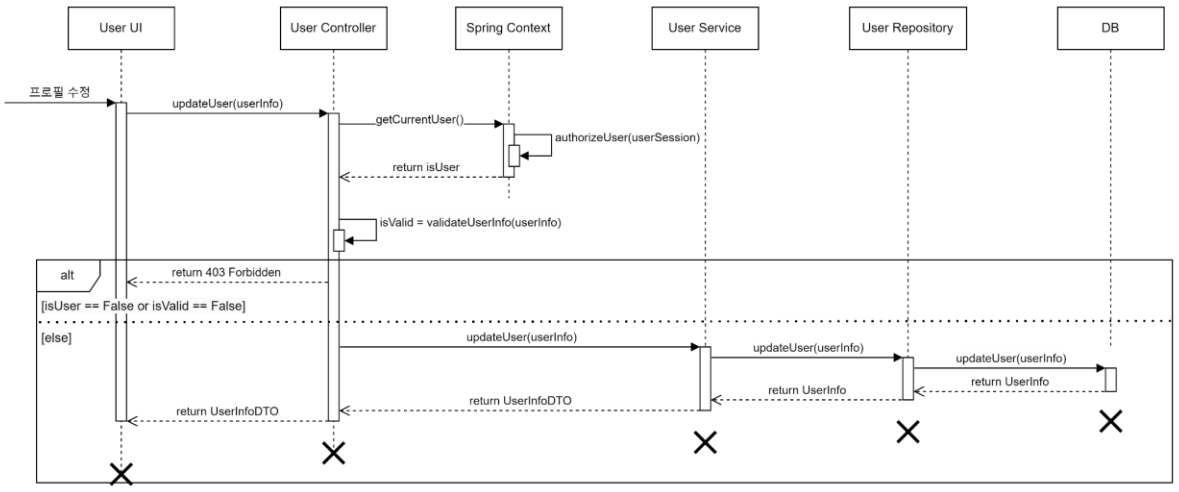


Figure 39: Sequence Diagram 14

사용자가 프로필 수정을 요청하면, User Controller 에서 한번의 검증을 거친 다음에 프로필 User Controller 는 사용자의 세션 정보를 검증하기 위해 Spring Context 에 있는 사용자 인증 메소드를 호출한다. 인증과 양식 검증에 성공하면 User Service 에 요청을 보내고 User

Repository 를 통해서 수정사항을 업데이트 한다. 바뀐 정보는 UI 에서 반환받을 수 있다. 실패시 403 Forbidden 을 반환받는다.

13.9 팔로우 목록 조회

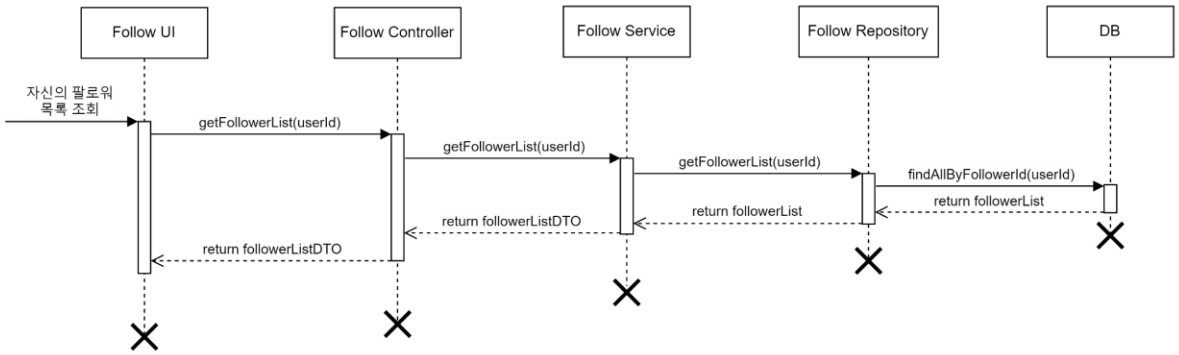


Figure 40: Sequence Diagram 15

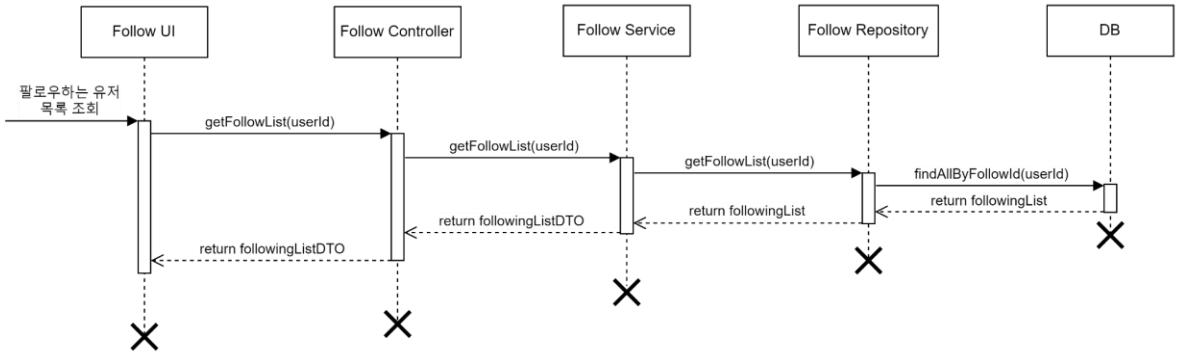


Figure 41: Sequence Diagram 16

자신이 팔로우한 사용자 목록 및 자신을 팔로우하는 사용자 목록을 조회할 수 있다.
회원이 팔로우 관련 목록 조회를 요청하면, Follow Controller 는 Follow Service 에 팔로우 관련 리스트를 요청한다. Follow Repository 를 통해 팔로우 관련 리스트를 한번에 얻고, 저장된 팔로우 리스트 정보를 DTO 형태로 변환하여 UI 로 다시 전달한다. 팔로우 정보는 모두에게 공개되어 별도의 검증을 필요로하지 않는다.

13.10 대시보드 조회

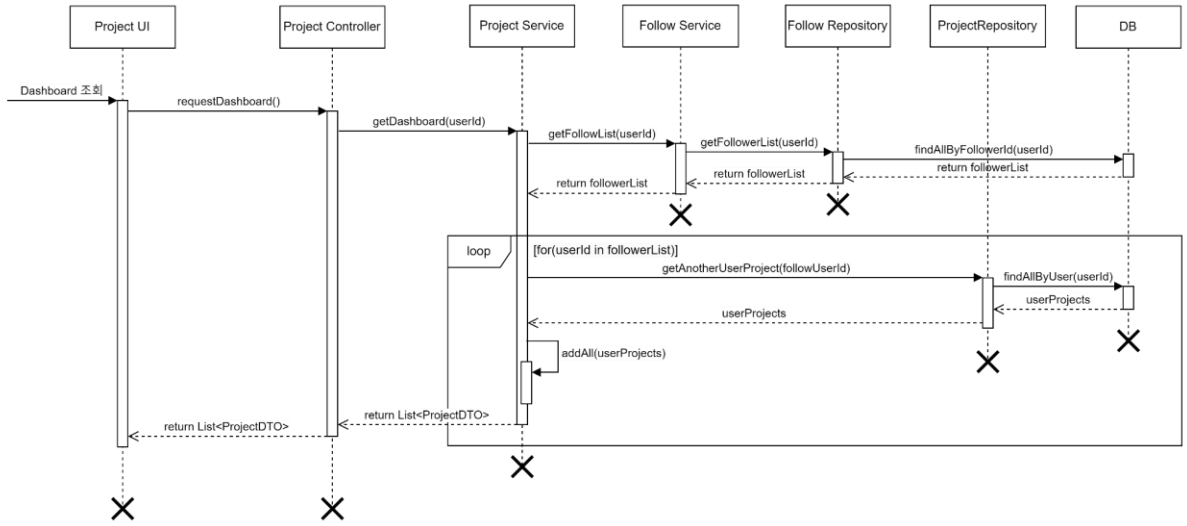


Figure 42: Sequence Diagram 17

대시보드에서는 팔로우한 유저의 프로젝트 목록 조회가 이루어진다. 대시보드 조회 시, Project Service 에서 팔로우한 사용자 목록을 불러오고 그 목록을 이용하여 그들의 프로젝트 정보를 모두 가져온다.

13.11 많이 사용된 라이브러리 목록

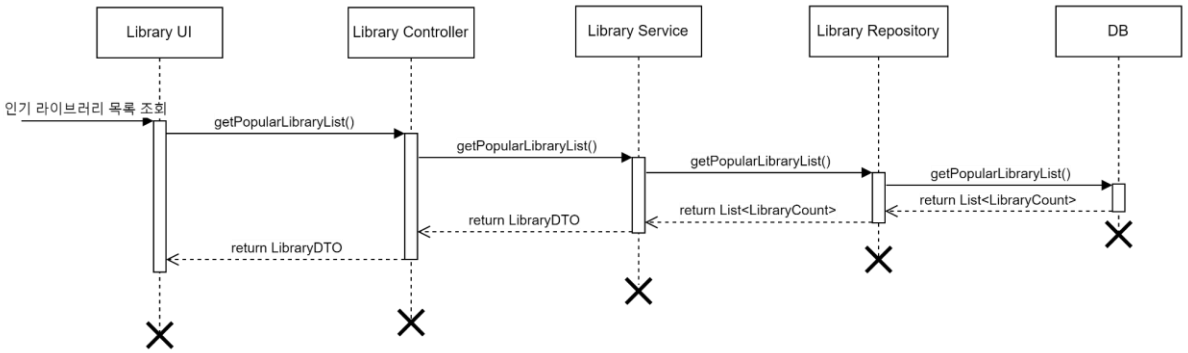


Figure 43: Sequence Diagram 18

Library service 에서 라이브러리가 추가될 때마다 데이터베이스에서 특정 라이브러리 이름의 등록 수가 증가하고, 삭제될 때마다 등록 수가 감소한다. 사용자는 그 등록 수에서 오름차순으로 목록을 정렬한 인기 라이브러리 목록을 DB 에서 가져와 조회할 수 있다.

14. Testing Plan & Result

14.1 Goals and Objectives

본 소프트웨어는 클라이언트 UI 에서 발생하는 다양한 use case scenarios 에 대해서 서버와의 통신이 원활하게 진행되어 요청이 처리되고, 페이지 전환이 불필요한 동작이나 큰 지연 없이 이루어져야한다. 또한 인증 시스템은 사용자가 권한이 없는 동작을 수행하는 것을 방지하고, 권한이 있는 동작에 대해서 오류 없이 사용자들의 요청을 처리해야한다.
다양한 test case 를 만들고 확인함으로써 필요한 기능의 수행이 가능하다는 것을 확인하고, 예외처리가 적절하게 되어서 시스템 이용의 불편이 생기지 않는다는 것을 확인하는 것이 이번 소프트웨어 테스트의 목표이다.

추가적으로 Non-Functional Requirements 에서도 언급했듯이 본 소프트웨어는 많은 사용자가 많은 데이터를 요청하더라도 서비스가 적절하게 작동해야만한다. 따라서 불러올 데이터가 많은 데이베이스 요청을 처리할 필요가 있는 서비스이므로 다량의 요청을 수월하게 수행할 수 있는지 검증해야한다.

14.2 Statement of Scope

검증 가능한 use case scenarios 를 기존 use case 에서 이끌어내어 요청이 성공하는 일반적인 경우를 먼저 테스트한다. 그런 다음에 사전 조건이나 동작 과정에서 발생하는 예외가 발생하는 경우에 대해서 추가 테스트를 진행하여 요구사항에 맞게 처리되는지 확인할 것이다. 이렇게 함으로써 프로젝트, 라이브러리, 사용자 정보, 팔로우 정보, 좋아요 정보의 복잡한 연관 관계가 정상적으로 작동하여 데이터가 처리되는지 확인하여 소프트웨어의 사용성(Usability)과 신뢰성(Reliability)을 증명하고자 한다.

그러나 주어진 기간이 부족과 경험의 부족으로 시스템의 성능을 더 세부적이고 구체적으로 측정하는 것에는 한계가 있을 수 있다. 따라서 이번 테스트에서는 스프링의 Apache JMeter 툴을 이용하여 시스템의 성능을 부분적으로나마 테스트 하는 것에 의의를 두었다.

14.3 Test Case

14.3.1 로그인/회원가입

TEST INFORMATION
Test Case ID: IT_01.1
Test Case Name: 회원가입 성공
For Use Case: 로그인/회원가입
Related Requirement IDs: UR_01, UR_02, SFR_01, SFR_02

Precondition: 사용자는 등록된 계정이 없어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 로그인 페이지에 접속한다.	로그인 페이지가 렌더링 되어야한다.
2	사용자가 구글 로그인 버튼을 클릭한다.	구글 로그인 페이지가 렌더링 되어야한다.
3	사용자가 구글 로그인을 완료한다.	사용자의 계정이 생성되어 서버에 저장되고 로그인 정보가 세션에 저장되며, 홈 페이지가 렌더링 되어야 한다.

Table 64: Test Case 01

TEST INFORMATION		
Test Case ID: IT_01.2		
Test Case Name: 로그인 성공		
For Use Case: 로그인/회원가입		
Related Requirement IDs: UR_01, UR_02, SFR_01, SFR_02		
Precondition: 사용자는 등록된 계정이 존재해야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 로그인 페이지에 접속한다.	로그인 페이지가 렌더링 되어야한다.
2	사용자가 구글 로그인 버튼을 클릭한다.	구글 로그인 페이지가 렌더링 되어야한다.
3	사용자가 구글 로그인을 완료한다.	사용자의 로그인 정보가 세션에 저장되고 홈 페이지가 렌더링 되어야 한다.

Table 65: Test Case 02

14.3.2 로그아웃

TEST INFORMATION		
Test Case ID: IT_02.1		
Test Case Name: 로그아웃 성공		
For Use Case: 로그아웃		
Related Requirement IDs: UR_04, SFR_06		
Precondition: 사용자는 시스템에 로그인이 되어 있어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 로그아웃 버튼을 클릭한다.	사용자의 로그인 정보가 세션에서 삭제되고 홈페이지가 렌더링 되어야 한다.

Table 66: Test Case 03

14.3.3 회원탈퇴

TEST INFORMATION		
Test Case ID: IT_03.1		
Test Case Name: 회원 탈퇴		
For Use Case: 회원 탈퇴		
Related Requirement IDs: UR_5 , SFR_07		
Precondition: 사용자는 시스템에 로그인이 되어 있어야 한다.		
TEST STEP		
#	Step	Expected Result

1	사용자가 회원탈퇴 버튼을 클릭한다.	사용자의 계정이 서버에서 삭제되고, 로그인 정보가 세션과 삭제되며, 홈페이지가 렌더링 되어야 한다.

Table 67: Test Case 04

14.3.4 프로젝트 생성

TEST INFORMATION		
Test Case ID: IT_04.1		
Test Case Name: 프로젝트 생성 성공		
For Use Case: 프로젝트 생성		
Related Requirement IDs: UR_06, SFR_08		
Precondition: 사용자는 시스템에 로그인이 되어 있어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 프로젝트 생성 버튼을 클릭한다.	프로젝트 이름, 해시태그, 설명, 관련 링크, 공개 여부를 입력할 수 있는 양식을 포함한 프로젝트 생성 페이지가 렌더링 되어야 한다.
2	사용자가 프로젝트 이름, 해시태그, 설명, 관련 링크를 입력 후 공개 여부를 선택한다.	프로젝트가 생성되어 서버에 저장되고 프로젝트 상세 페이지가 렌더링 되어야 한다.

Table 68: Test Case 05

TEST INFORMATION
Test Case ID: IT_04.2
Test Case Name: 프로젝트 필수 정보 누락
For Use Case: 프로젝트 생성

Related Requirement IDs: UR_06, SFR_08		
Precondition: 사용자는 시스템에 로그인이 되어 있어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 프로젝트 생성 버튼을 클릭한다.	프로젝트 이름, 해시태그, 설명, 관련 링크, 공개 여부를 입력받는 양식이 포함된 프로젝트 생성 페이지가 렌더링 되어야 한다.
2	사용자가 프로젝트 제목이나 설명을 누락 후 완료를 클릭한다.	프로젝트 필수 정보 누락 오류 메시지가 화면에 표시 되어야 한다.

Table 69: Test Case 06

14.3.5 프로젝트 조회

TEST INFORMATION		
Test Case ID: IT_05.1		
Test Case Name: 프로젝트 조회 성공		
For Use Case: 프로젝트 조회		
Related Requirement IDs: UR_09, SFR_12		
Precondition: None		
TEST STEP		
#	Step	Expected Result
1	사용자가 프로젝트를 클릭한다.	선택된 프로젝트의 등록 유저의 이름, 해시태그, 설명, 관련 링크, 라이브러리 목록이 포함된 프로젝트 상세 페이지가 렌더링 되어야 한다.
2	사용자가 라이브러리 토글을 클릭한다.	선택된 라이브러리의 이름, 해시태그, 버전, 사용 사례가 렌더링 되어야 한다.

Table 70: Test Case 07

14.3.6 프로젝트 수정

TEST INFORMATION		
Test Case ID: IT_06.1		
Test Case Name: 프로젝트 수정 성공		
For Use Case: 프로젝트 수정		
Related Requirement IDs: UR_10, SFR_12, SFR_13		
Precondition: 사용자는 프로젝트 소유자 이어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 프로젝트 수정 버튼을 클릭한다.	프로젝트의 이름, 해시태그, 설명, 관련 링크 양식이 포함된 프로젝트 수정 페이지가 렌더링 되어야 한다.
2	사용자가 양식에 있는 정보들을 수정 후 완료 버튼을 클릭한다.	선택된 라이브러리의 이름, 해시태그, 버전, 사용 사례가 렌더링 되어야 한다.

Table 71: Test Case 08

TEST INFORMATION
Test Case ID: IT_06.2
Test Case Name: 프로젝트 수정 실패
For Use Case: 프로젝트 수정
Related Requirement IDs: UR_10, SFR_12, SFR_13
Precondition: 사용자는 프로젝트 소유자 이어야 한다.
TEST STEP

#	Step	Expected Result
1	사용자가 프로젝트 수정 버튼을 클릭한다.	프로젝트의 이름, 해시태그, 설명, 관련 링크 양식이 포함된 프로젝트 수정 페이지가 렌더링 되어야 한다.
2	사용자가 프로젝트 제목이나 설명을 누락 후 완료 버튼을 클릭한다.	수정된 프로젝트 정보를 서버에 업데이트 후 프로젝트 상세 페이지가 렌더링 되어야 한다.

Table 72: Test Case 09

TEST INFORMATION		
Test Case ID: IT_06.3		
Test Case Name: 프로젝트 삭제 성공		
For Use Case: 프로젝트 수정		
Related Requirement IDs: UR_11, SFR_12, SFR_13		
Precondition: 사용자는 프로젝트 소유자 이어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 프로젝트 삭제 버튼을 클릭한다.	확인과 취소 버튼을 포함한 삭제 재확인 창이 렌더링 되어야한다.
2	사용자는 삭제 재확인 버튼을 클릭한다.	프로젝트를 서버에서 삭제 후 프로젝트 목록 페이지가 렌더링 되어야 한다.

Table 73: Test Case 10

14.3.7 라이브러리 생성

TEST INFORMATION
Test Case ID: IT_07.1
Test Case Name: 라이브러리 생성 성공

For Use Case: 라이브러리 생성		
Related Requirement IDs: UR_07, SFR_09		
Precondition: 사용자는 시스템에 로그인이 되어 있고 프로젝트 소유자이여야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 라이브러리 생성 버튼을 클릭한다.	라이브러리 이름, 버전, 해시태그, 사용 사례 및 설명 양식이 포함된 라이브러리 생성 페이지가 렌더링 되어야 한다.
2	사용자가 라이브러리 이름, 버전, 해시태그, 사용 사례 및 설명을 입력 후 완료를 클릭한다.	라이브러리가 생성되어 서버에 저장되고 프로젝트 상세 페이지가 렌더링 되어야 한다.

Table 74: Test Case 11

TEST INFORMATION		
Test Case ID: IT_07.2		
Test Case Name: 라이브러리 필수 정보 누락		
For Use Case: 라이브러리 생성		
Related Requirement IDs: UR_07, SFR_09		
Precondition: 사용자는 시스템에 로그인이 되어 있고 프로젝트 소유자이여야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 라이브러리 생성 버튼을 클릭한다.	라이브러리 이름, 버전, 해시태그, 사용 사례 및 설명 양식이 포함된 라이브러리 생성 토글이 렌더링 되어야 한다.
2	사용자가 라이브러리 이름, 버전, 사용 사례 및 설명을 누락된 상태로	라이브러리 필수 정보 누락 오류 메시지가 화면에 표시 되어야 한다.

	입력 후 완료를 클릭한다.	

Table 75: Test Case 12

14.3.8 라이브러리 수정

TEST INFORMATION		
Test Case ID: IT_08.1		
Test Case Name: 라이브러리 수정 성공		
For Use Case: 라이브러리 수정		
Related Requirement IDs: UR_10, SFR_12, SFR_13		
Precondition: 사용자는 프로젝트 소유자 이어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 라이브러리 수정 버튼을 클릭한다.	라이브러리 이름, 버전, 해시태그, 사용 사례를 입력하는 양식이 포함된 라이브러리 수정 토글이 렌더링 되어야 한다.
2	사용자가 양식에 있는 정보들을 수정 후 완료 버튼을 클릭한다.	수정된 라이브러리 정보를 서버에 업데이트 후 라이브러리 상세 토글이 렌더링 되어야 한다.

Table 76: Test Case 13

TEST INFORMATION
Test Case ID: IT_08.2
Test Case Name: 라이브러리 수정 실패
For Use Case: 라이브러리 수정
Related Requirement IDs: UR_10, SFR_12, SFR_13

Precondition: 사용자는 프로젝트 소유자 이어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 라이브러리 수정 버튼을 클릭한다.	라이브러리 이름, 버전, 해시태그, 사용 사례 양식이 포함된 라이브러리 수정 토글이 렌더링 되어야 한다.
2	사용자가 라이브러리 이름, 버전, 사용 사례 및 설명을 누락 후 완료 버튼을 클릭한다.	라이브러리 필수 정보 누락 오류 메시지가 화면에 표시 되어야 한다.

Table 77: Test Case 14

TEST INFORMATION		
Test Case ID: IT_06.3		
Test Case Name: 라이브러리 삭제 성공		
For Use Case: 프로젝트 수정		
Related Requirement IDs: UR_11, SFR_12, SFR_13		
Precondition: 사용자는 프로젝트 소유자 이어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 라이브러리 삭제 버튼을 클릭한다.	확인과 취소 버튼을 포함한 삭제 재확인 창이 렌더링 되어야한다.
2	사용자가 삭제 확인 버튼을 클릭한다.	라이브러리를 서버에서 삭제 후 프로젝트 상세 페이지가 렌더링 되어야 한다.

Table 78: Test Case 15

14.3.9 공유 프로젝트 검색

TEST INFORMATION

Test Case ID: IT_09.1		
Test Case Name: 공유 프로젝트 검색 성공		
For Use Case: 공유 프로젝트 검색		
Related Requirement IDs: UR_12, SFR_14		
Precondition: None		
TEST STEP		
#	Step	Expected Result
1	사용자가 검색 창에 키워드를 입력한다.	키워드와 일치하는 프로젝트 이름, 또는 해시태그를 가진 프로젝트 목록 페이지가 렌더링 되어야 한다.

Table 79: Test Case 16

14.3.10 프로필 정보 수정

TEST INFORMATION		
Test Case ID: IT_10.1		
Test Case Name: 프로필 정보 수정 성공		
For Use Case: 프로필 수정		
Related Requirement IDs: UR_05, SFR_07		
Precondition: 사용자는 로그인하여 자신의 프로필 페이지에 있어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 수정 버튼을 클릭한다.	사용자 이름, 프로필 이미지, 관련 링크를 수정할 수 있는 프로필 수정 팝업 페이지가 렌더링 된다.

2	사용자가 사용자 이름, 프로필 이미지, 관련 링크를 수정한다.	정상적으로 내용이 입력된다.
3	사용자가 완료 버튼을 누른다.	수정 완료 메세지가 뜨고, 프로필 페이지로 이동한다. 수정한 내용과 동일한 내용으로 프로필 정보가 갱신된다.

Table 80: Test Case 17

TEST INFORMATION		
Test Case ID: IT_10.2		
Test Case Name: 프로필 정보 수정 실패		
For Use Case: 프로필 수정		
Related Requirement IDs: UR_05, SFR_07		
Precondition: 사용자는 로그인하여 자신의 프로필 페이지에 있어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 수정 버튼을 클릭한다.	사용자 이름, 프로필 이미지, 관련 링크를 수정할 수 있는 프로필 수정 팝업 페이지가 렌더링 된다.
2	사용자가 유효하지 않은 양식을 기입한다.	정상적으로 내용이 입력된다.
3	사용자가 완료 버튼을 누른다.	에러 메세지가 뜨고, 기존 창을 유지한다.

Table 81: Test Case 18

TEST INFORMATION
Test Case ID: IT_10.3
Test Case Name: 프로필 이미지 업로드 실패

For Use Case: 프로필 수정		
Related Requirement IDs: UR_05, SFR_07		
Precondition: 사용자는 로그인하여 자신의 프로필 페이지에 있어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 수정 버튼을 클릭한다.	사용자 이름, 관련 링크, 자주 사용하는 라이브러리를 수정할 수 있는 프로필 수정 팝업 페이지가 렌더링 된다.

Table 82: Test Case 19

14.3.11 프로젝트 좋아요

TEST INFORMATION		
Test Case ID: IT_11.1		
Test Case Name: 프로젝트 좋아요 등록 성공		
For Use Case: 프로젝트 좋아요 등록		
Related Requirement IDs: UR_13, SFR_15		
Precondition: 사용자는 로그인하여 좋아요하지 않은 특정 프로젝트를 조회하고 있어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 좋아요 버튼을 누른다.	해당 프로젝트가 좋아요 상태로 변한다.

Table 83: Test Case 20

TEST INFORMATION

Test Case ID: IT_11.2		
Test Case Name: 프로젝트 좋아요 등록 실패		
For Use Case: 프로젝트 좋아요 등록		
Related Requirement IDs: UR_13, SFR_15		
Precondition: 사용자는 로그인하여 좋아요한 특정 프로젝트를 조회하고 있어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 좋아요 버튼을 누른다.	해당 프로젝트의 좋아요 상태가 해제된다.

Table 84: Test Case 21

14.3.12 유저 팔로우

TEST INFORMATION		
Test Case ID: IT_12.1		
Test Case Name: 사용자 팔로우 성공		
For Use Case: 사용자 팔로우 및 언팔로우		
Related Requirement IDs: RS_15, SFR_17		
Precondition: 사용자는 시스템에 로그인 되어 있어야 한다. 특정 사용자는 팔로우하지 않은 상태이어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 특정 사용자의 프로필을 클릭한다.	해당 사용자의 프로필 페이지로 이동한다.
2	특정 사용자의 프로필 페이지에서 팔로우 버튼을 클릭한다.	해당 사용자가 팔로우 되어 팔로우 상태가 렌더링된다.

Table 85: Test Case 22

TEST INFORMATION		
Test Case ID: IT_12.2		
Test Case Name: 사용자 언팔로우 성공		
For Use Case: 사용자 팔로우 및 언팔로우		
Related Requirement IDs: RS_15, SFR_17		
Precondition: 사용자는 시스템에 로그인 되어 있어야 한다. 특정 사용자는 팔로우한 상태이어야 한다.		
TEST STEP		
#	Step	Expected Result
1	사용자가 특정 사용자의 프로필을 클릭한다.	해당 사용자의 프로필 페이지로 이동한다.
2	특정 사용자의 프로필 페이지에서 팔로우 버튼을 클릭한다.	해당 사용자가 언팔로우 되어 언팔로우 상태가 렌더링된다.

Table 86: Test Case 23

14.3.13 프로필 조회

TEST INFORMATION
Test Case ID: IT_13.1
Test Case Name: 자신의 프로필 조회 성공
For Use Case: 프로필 조회
Related Requirement IDs: RS_16, SFR_18
Precondition: 사용자가 시스템에 로그인되어 있어야 한다.
TEST STEP

#	Step	Expected Result
1	사용자가 “My Profile” 카탈로그를 클릭한다.	사용자의 프로필 페이지로 이동한다. 사용자의 이름, 관련 링크, 자주 사용하는 라이브러리가 렌더링되고, 수정 버튼이 렌더링 된다.

Table 87: Test Case 24

TEST INFORMATION		
Test Case ID: IT_13.2		
Test Case Name: 타인의 프로필 조회 성공		
For Use Case: 프로필 조회		
Related Requirement IDs: RS_16, SFR_18		
Precondition: None		
TEST STEP		
#	Step	Expected Result
1	사용자가 다른 사용자의 프로필 사진을 클릭한다.	해당 사용자 이름, 관련 링크, 자주 사용하는 라이브러리를 조회할 수 있는 페이지가 렌더링 된다.

Table 88: Test Case 25

14.3.14 관심 프로젝트 검색

TEST INFORMATION
Test Case ID: IT_14.1
Test Case Name: 관심 프로젝트 조회 성공
For Use Case: 관심 프로젝트 조회
Related Requirement IDs: RS_14, SFR_16

Precondition: 사용자는 시스템에 로그인 되어 있어야하고, 적어도 1 개 이상의 프로젝트를 좋아요 등록해야한다.		
TEST STEP		
#	Step	Expected Result
1	회원이 “Favorites” 카탈로그를 클릭한다.	관심 프로젝트 목록 페이지로 이동하여, 좋아요한 프로젝트 목록을 조회한다.
2	회원이 목록 중에서 임의의 프로젝트를 클릭한다.	해당 프로젝트 상세 조회 페이지로 이동하여, 프로젝트 정보를 조회한다.

Table 89: Test Case 26

TEST INFORMATION		
Test Case ID: IT_14.2		
Test Case Name: 빈 관심 프로젝트 목록 조회		
For Use Case: 관심 프로젝트 조회		
Related Requirement IDs: RS_14, SFR_16		
Precondition: 사용자는 시스템에 로그인 되어 있어야하고, 좋아요한 프로젝트가 존재하지 않아야한다.		
TEST STEP		
#	Step	Expected Result
1	회원이 “Favorites” 카탈로그를 클릭한다.	관심 프로젝트 목록 페이지로 이동하지만, 비어있는 프로젝트 목록이 렌더링된다.

Table 90: Test Case 27

14.3.15 팔로잉/팔로워 목록 조회

TEST INFORMATION
Test Case ID: IT_15.1

Test Case Name: 팔로잉 목록 조회 및 프로필 조회		
For Use Case: 팔로잉 목록 조회		
Related Requirement IDs: RS_17, SFR_19		
Precondition: 사용자는 시스템에 로그인되어 있어야한다.		
TEST STEP		
#	Step	Expected Result
1	회원이 “Following/Follower” 카탈로그를 클릭한다.	기본값인 팔로워 목록이 먼저 표시되고, 각 회원마다 팔로우 및 언팔로우를 할 수 있는 버튼이 제공된다.
2	회원이 “Following” 탭을 클릭한다.	팔로잉 목록이 표시된다.
3	회원이 팔로잉 목록에서 사용자 프로필을 클릭한다.	프로필 조회 페이지로 넘어간다.

Table 91: Test Case 28

TEST INFORMATION		
Test Case ID: IT_15.2		
Test Case Name: 팔로워 목록 조회 및 언팔로우		
For Use Case: 팔로워 목록 조회		
Related Requirement IDs: RS_17, SFR_19		
Precondition: 사용자는 시스템에 로그인되어 있어야한다.		
TEST STEP		
#	Step	Expected Result
1	회원이 “Following/Follower” 카탈로그를 클릭한다.	기본값인 팔로워 목록이 먼저 표시되고, 각 회원마다 팔로우 및 언팔로우를 할 수 있는

		버튼이 제공된다.
2	회원이 팔로워 목록에서 특정 유저에 언팔로우 버튼을 클릭한다.	특정 유저가 언팔로우되고 언팔로우 버튼이 팔로우 버튼으로 바뀐다.

Table 92: Test Case 29

14.3.16 대시보드 조회

TEST INFORMATION		
Test Case ID: IT_16.1		
Test Case Name: 대시보드 목록 조회 성공		
For Use Case: 대시보드 조회		
Related Requirement IDs: RS_18, SFR_20		
Precondition: 사용자는 시스템에 로그인되어 있어야한다. 사용자가 팔로우한 유저 중에 공개된 프로젝트를 소유하고 있는 사용자가 한 명 이상 있어야한다.		
TEST STEP		
#	Step	Expected Result
1	회원이 메인 홈페이지에 접근한다.	팔로우한 사용자의 프로젝트 목록이 렌더링된다.
2	프로젝트 목록의 특정 프로젝트를 클릭한다.	프로젝트 조회 페이지로 이동하여 프로젝트 내용이 렌더링된다.

Table 93: Test Case 30

TEST INFORMATION
Test Case ID: IT_16.2
Test Case Name: 대시보드 목록 조회 실패 1
For Use Case: 대시보드 조회
Related Requirement IDs: RS_18, SFR_20

Precondition: 사용자는 시스템에 로그인되어 있어야한다. 사용자는 팔로우한 사용자가 없어야한다.		
TEST STEP		
#	Step	Expected Result
1	회원이 메인 홈페이지에 접근한다.	프로젝트 목록이 렌더링되지 않고 비어있다.

Table 94: Test Case 31

TEST INFORMATION
Test Case ID: IT_16.3
Test Case Name: 대시보드 목록 조회 실패 2
For Use Case: 대시보드 조회
Related Requirement IDs: RS_18, SFR_20
Precondition: 사용자는 시스템에 로그인되어 있어야한다. 사용자는 팔로우한 사용자 모두가 프로젝트 목록이 없어야한다.

TEST STEP		
#	Step	Expected Result
1	회원이 메인 홈페이지에 접근한다.	프로젝트 목록이 렌더링되지 않고 비어있다.

Table 95: Test Case 32

14.3.17 인기 라이브러리 목록 조회

TEST INFORMATION
Test Case ID: IT_17.1
Test Case Name: 인기 라이브러리 목록 조회
For Use Case: 인기 라이브러리 조회

Related Requirement IDs: RS_19, SFR_21		
Precondition: None		
TEST STEP		
#	Step	Expected Result
1	사용자가 메인 페이지에 접근한다.	사이드 바에서 자주 사용되는 라이브러리가 표시되도록 렌더링한다.

Table 96: Test Case 33

14.3.18 동시 접속자 동시 요청

TEST INFORMATION		
Test Case ID: IT_18.1		
Test Case Name: 동시 접속자 동시 요청(대시보드)		
For Use Case: 대시보드 조회		
Related Requirement IDs: RS_18, SFR_20, NFR_03		
Precondition: 대시보드에서 가져올 프로젝트가 총 40 개가 있다. Apache JMeter 테스팅 툴에서 가상의 쓰레드 100 개를 둔다.		
TEST STEP		
#	Step	Expected Result
1	각 쓰레드에서 10 번의 대시보드 조회 요청을 보낸다.	대시보드 정보를 모든 쓰레드가 10 번 반복하여 받는다.

Table 97: Test Case 34

TEST INFORMATION
Test Case ID: IT_18.2
Test Case Name: 동시 접속자 동시 요청(프로젝트 조회)

For Use Case: 프로젝트 조회		
Related Requirement IDs: RS_18, SFR_20, NFR_03		
Precondition: Apache JMeter 테스팅 툴을 이용해 가상의 쓰레드 100 개를 둔다.		
TEST STEP		
#	Step	Expected Result
1	각 쓰레드에서 10 번의 특정 프로젝트 상세 정보를 조회 요청을 보낸다.	특정 프로젝트 상세 정보를 모든 쓰레드가 10 번 반복하여 받는다.

Table 98: Test Case 35

14.3.19 시간 내 요청 처리

TEST INFORMATION		
Test Case ID: IT_19.1		
Test Case Name: 시간 내 렌더링 완료(최소 조건)		
For Use Case: 전체		
Related Requirement IDs: RS_01 ~ RS_19, SFR_01~SFR_21, NFR_02		
Precondition: 조회 할 프로젝트, 라이브러리, 좋아요, 팔로우 관련 데이터가 1 개 이상 존재한다.		
TEST STEP		
#	Step	Expected Result
1	모든 테스트를 실행하면서 실행 시간을 확인한다.	모든 테스트케이스에 대하여 실행 시간이 3 초 이내이다.

Table 99: Test Case 36

TEST INFORMATION

Test Case ID: IT_19.2		
Test Case Name: 시간 내 렌더링 완료(실전 조건)		
For Use Case: 전체		
Related Requirement IDs: RS_01 ~ RS_19, SFR_01~SFR_21, NFR_02		
Precondition: 조회할 프로젝트 및 라이브러리 데이터가 1000 개 이상, 팔로워, 팔로잉 회원이 1000 명 이상, 좋아요한 프로젝트 100 개 이상		
TEST STEP		
#	Step	Expected Result
1	모든 use case 를 실행하면서 실행 시간을 확인한다.	모든 테스트케이스에 대하여 실행 시간이 3 초 이내이다.

Table 100: Test Case 37

14.4 Test Results

14.4.1 Result Table

Test Case Number	Test Case Name	Test Case Result	Comments
1.1	회원가입 성공	pass	
1.2	로그인 성공	pass	
2.1	로그아웃 성공	pass	
3.1	회원탈퇴 성공	pass	
4.1	프로젝트 생성 성공	pass	
4.2	프로젝트 생성 실패	pass	
5.1	프로젝트 조회 성공	pass	
6.1	프로젝트 수정 성공	pass	
6.2	프로젝트 수정 실패	pass	
6.3	프로젝트 삭제 성공	pass	
7.1	라이브러리 생성 성공	pass	

7.2	라이브러리 생성 실패	pass	
8.1	라이브러리 수정 성공	pass	
8.2	라이브러리 수정 실패	pass	
8.3	라이브러리 삭제 성공	pass	
9.1	공유 프로젝트 검색 성공	pass	
9.2	자신의 프로젝트 검색 성공	pass	
10.1	프로필 정보 수정 성공	pass	
10.2	프로필 정보 수정 실패	pass	
10.3	프로필 이미지 업로드 실패	pass	
11.1	프로젝트 좋아요 등록 성공	pass	
11.2	프로젝트 좋아요 해제 성공	pass	
11.3	프로젝트 좋아요 검색 성공	pass	
12.1	유저 팔로우 성공	pass	
12.2	유저 언팔로우 성공	pass	
13.1	자신의 프로필 조회 성공	pass	
13.2	타인의 프로필 조회 성공	pass	
14.1	관심 프로젝트 목록 검색 성공	pass	
14.2	빈 프로젝트 목록 조회(관심)	pass	
15.1	팔로잉 목록 조회 및 프로필 조회	pass	
15.2	팔로워 목록 조회 및 언팔로우	pass	
16.1	대시보드 검색 성공	pass	
16.2	대시보드 검색 실패 1	pass	
16.3	대시보드 검색 실패 2	pass	
17. 1	인기 라이브러리 목록 조회	pass	
18.1	동시 접속자 동시 요청(대시보드)	pass	
18.2	동시 접속자 동시 요청(프로젝트 조회)	pass	

19.1	시간 내 요청 처리(최소 조건)	pass	
19.2	시간 내 요청 처리(실전 조건)	pending	테스팅 난이도와 투자할 자원의 부족으로 보류

Table 101: Test Result 01

전체 테스트	39	
테스트 통과	38	98%
테스트 실패	0	0%
테스트 대기	1	2%
테스트 이행률	38/39	98%
테스트 성공률	**38/38**	**100%**

Table 102: Test Result 02

14.4.2 Validation Screenshot

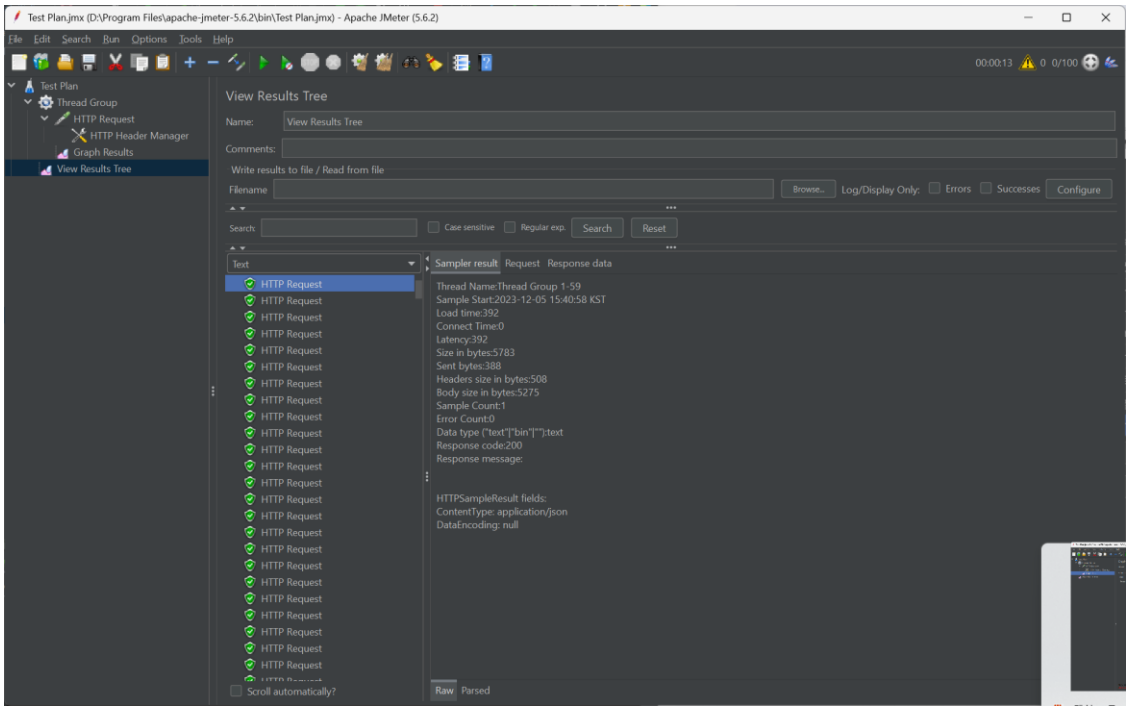


Figure 44: Validation Result 01

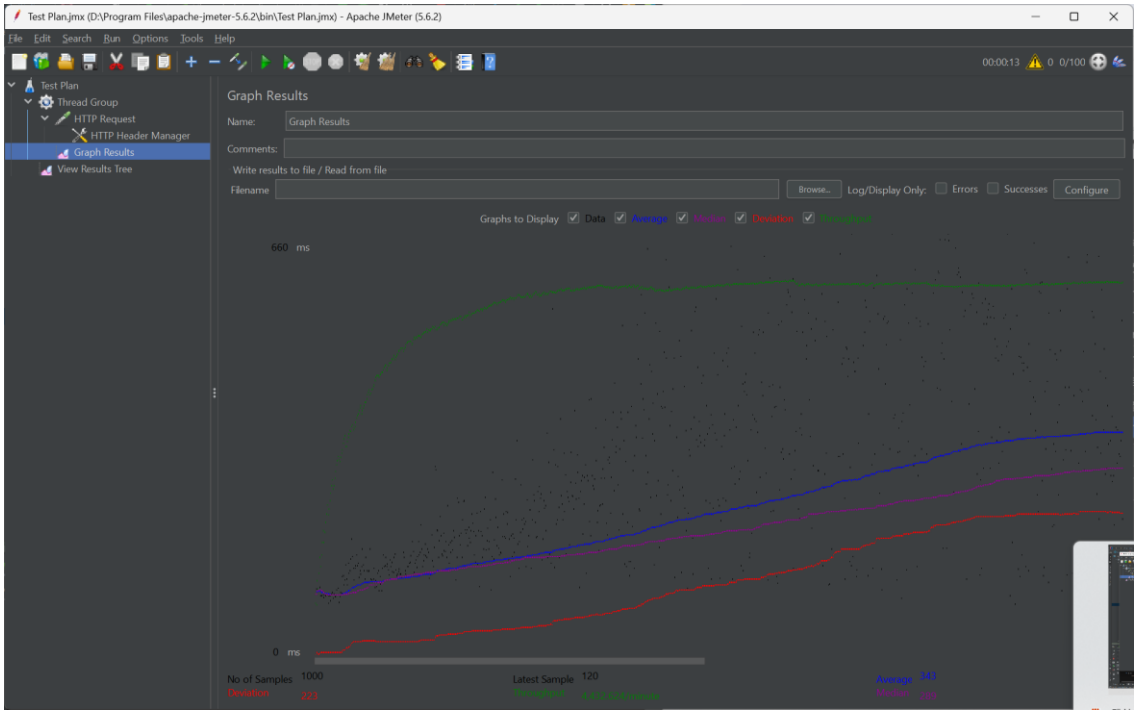


Figure 45: Validation Result 02

15. Implementation Details

15.1 Implementation

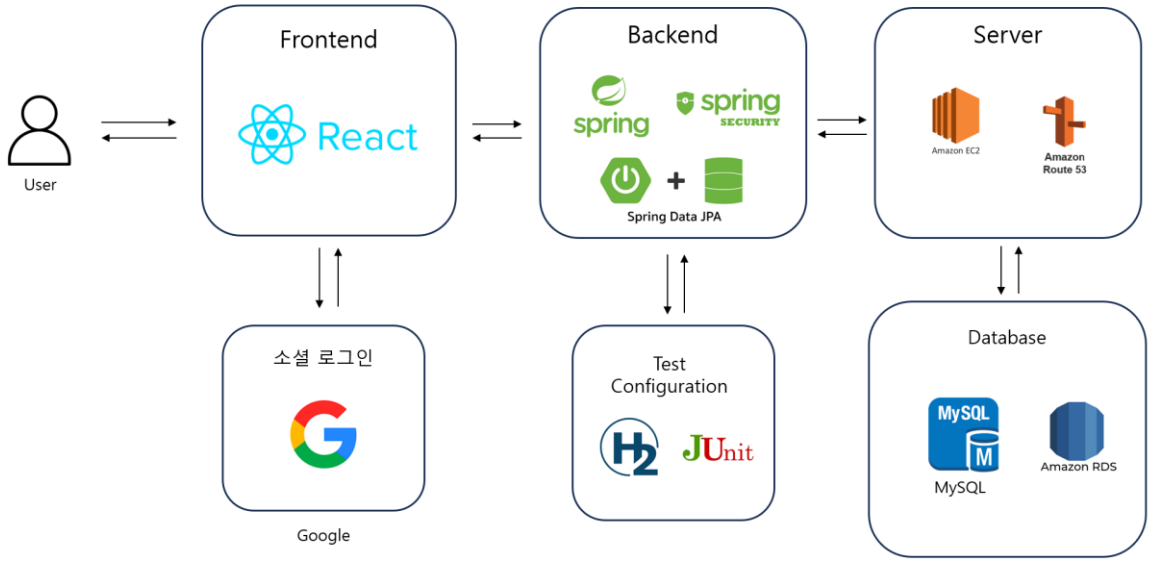


Figure 46: System Architecture

15.1.1 Language & Library

- **Frontend**
  - **HTML, CSS, JavaScript**
    - 웹 프론트엔드를 구성하는 기본 언어이다.
  - **React**
    - UI 를 만들기 위한 JavaScript 의 라이브러리이다.
    - 캡슐화된 컴포넌트를 조합하여 복잡한 UI 를 구성할 수 있다.
  - **Axios**
    - Promise 기반 HTTP 클라이언트 라이브러리이다.
    - HTTP 통신을 할 때 JSON 형식으로 parsing 하는 수고를 덜 수 있고, 모듈화하여 사용할 수 있어 유지/보수에 용이하다.
  - **Recoil**
    - React 의 전역상태관리 라이브러리이다.
    - 전역상태관리 라이브러리로 많이 사용하는 Redux 에 비해 boilerplate 코드가 적고 사용하기 쉽다.
  - **Material UI**
    - React 의 UI 라이브러리이다.
    - 미리 디자인되고 만들어져 있는 컴포넌트를 손쉽게 사용할 수 있다.
    - 컴포넌트 커스텀화가 가능하다.

- **Backend**
  - **Java**
    - 백엔드를 구성하는 기본 언어이다.
  - **Spring Boot**
    - 자바 언어를 이용하여 웹 애플리케이션 서버를 만드는 것을 도와주는 프레임워크이다. MVC 패턴과 Restful API 통신, 객체 라이프사이클 관리 등을 쉽게 해준다.
  - **Spring Data JPA**
    - Java Persistence API (JPA)를 사용하여 ORM (Object Relation Mapping)을 지원한다.
    - 기존 데이터베이스에서는 못하는 객체로 필드를 관리하는 것을 가능하도록 해준다.
  - **Spring Security**
    - Spring 기반의 애플리케이션에서 보안 관련 기능을 제공하는 프레임워크이다.
    - Spring Security 를 통해 OAuth2 (소셜 로그인) 과 로그인한 유저에 대해서만 엔드포인트 요청을 허용하도록 시큐리티 필터를 통해 보안을 높일 수 있었으며, 로그인한 유저의 컨텍스트를 관리한다.
  - **Lombok**
    - 자바의 어노테이션과 Argument Resolver 를 활용해 코드를 런타임이나 컴파일 시간에 삽입해주는 라이브러리이다. 코드의 가독성을 높이기 위해 사용했다.
  - **MySQL**
    - 관계형 데이터베이스 관리 시스템 (RDBMS) 중 하나로, 웹 애플리케이션의 데이터를 저장할 데이터베이스로 활용한다.
  - **H2**
    - H2 데이터베이스는 자바 기반의 경량 관계형 데이터베이스 관리 시스템이다.
    - 테스트를 하기 위해 실제 데이터베이스를 연결하여 사용하기에는 무겁기 때문에, H2 인메모리 DB 를 사용해 테스트 퍼포먼스를 향상시켰다.
  - **JUnit**

        - 자바 기반 프로그래밍 언어를 위한 테스트 프레임워크로, 기능 단위 테스트를 하기 위해 사용했다. 테스트를 통해 코드의 품질을 향상시키고, 안정성을 보장하는데 도움을 줬다.
    - **AmazonS3Config**
        - Amazon 의 S3 를 이용하기 위해 아마존에서 제공하는 라이브러리이다. Amazon 의 S3 는 파일을 저장할 수 있는 저장 공간을 제공해주므로, 유저의 프로필 사진을 저장하는데 사용했다.

15.1.2 Authentication

- **OAuth 2.0**
    - Google
        - 구글 인증을 통해 서비스 회원가입 및 로그인이 가능하다.
        - Spring Security 를 사용하여 Google 로그인을 통한 사용자 인증을 구현했다.
- **JWT**
    - Access Token
        - JWT 의 access token 을 사용하여 사용자 인증을 진행한다.
        - 클라이언트단에서 HTTP 통신을 할 때 Header 에 access token 을 포함시키고, 서버의 Authentication Filter 에서 access token 으로 사용자 인증을 진행한다.
        - 세션 단위에서 로그인 상태를 지속시키기 위해 access token 을 클라이언트의 session storage 에 저장한다.
- **Spring Security**
    - Security Filter
        - 유저의 로그인한 상태인지 검사하는 필터를 만든다. 로그인한 유저인지 여부에 따라 엔드포인트 접근에 대한 제한을 쉽게 둘 수 있다. 또한 로그인한 유저를 Security Context 로 관리하여, 각 요청에 대해 동일한 컨텍스트를 볼 수 있도록 도와준다.

15.1.3 Deployment

- **AWS**
    - EC2

        - Amazon 에서 제공하는 클라우드 컴퓨팅 서비스이다. EC2 를 사용하여 서비스를 배포하여, 웹에서 도메인으로 접근이 가능하도록 했다.
          종류는 무료로 사용 가능한 t2 micro 를 이용했다
    - RDS
        - 데이터베이스를 원격 서버에 올리기 위해 사용했다. RDS 는 데이터베이스를 위한 클라우드 컴퓨터이기 때문에 설치와 사용이 쉽다는 장점이 있으며, EC2 를 t2 micro 를 사용하기 때문에, 데이터베이스 접근에 대한 부담을 덜기 위해 사용했다.
    - Amazon S3
        - 파일 유형의 데이터를 저장하기 위해 사용했다. 서비스 중 유저 프로필 이미지 업로드와 조회할 때 사용하기 위해 이용했다.

15.1.4 Communication

- **Development**
    - GitHub
        - 코드 버전 관리를 위해 사용했다.
    - Swagger
        - 원할한 협업을 위해 Swagger Open API 3.0 을 이용하여 엔드포인트를 볼 수 있으며, 테스팅 해볼 수 있도록 구성했다.
- **Management**
    - Notion
        - 프로젝트의 전반적인 계획, 진행, 설계를 위해 사용했다.

15.2 API Specification

15.2.1 User Controller

Type	URI	Description
POST	/api/user/update	유저 정보 업데이트
GET	/api/user/usedLibrary/{userId}	특정 유저가 자주 사용하는 라이브러리 목록 조회
GET	/api/user/usedLibrary/my	자신이 자주 사용하는 라이브러리 목록 조회
GET	/api/user/info	현재 로그인한 유저의 정보 조회
GET	/api/user/info/{userId}	다른 유저의 상세 정보 조회 + 팔로우하는 유저인지 확인 가능
GET	/api/user/favorites/{pageNumber}	현재 로그인한 유저가 즐겨찾기 한 프로젝트들 조회
DELETE	/api/user/delete	회원 탈퇴

Table 103: API Specification 01

회원 정보와 관련한 요청을 위한 API 이다. 다른 회원의 정보 또한 요청할 수 있고, 나의 팔로우 정보 및 좋아요 정보를 이용하여 필요한 정보를 제공받을 수 있다.

15.2.2 Library Controller

Type	URI	Description
POST	/api/projects/{projectId}/libraries	프로젝트에 라이브러리 정보 추가
GET	/api/projects/libraries/{libraryId}	라이브러리 상세 정보 조회
DELETE	/api/projects/libraries/{libraryId}	라이브러리 삭제
PATCH	/api/projects/libraries/{libraryId}	라이브러리 정보 수정

Table 104: API Specification 02

서비스의 핵심인 라이브러리에 관한 엔드포인트 API 이다. 라이브러리 등록은 프로젝트에 의존하기 때문에 프로젝트 고유 id 를 파라미터로 입력해야한다. 이외에도 라이브러리 검색이나 인기 라이브러리, 유저가 자주 사용하는 라이브러리에 관한 로직은 프로젝트와 유저 컨트롤러에 속해있다.

15.2.3 Project Controller

Type	URI	Description
GET	/api/project	모든 공개 프로젝트 조회
POST	/api/project	프로젝트 등록
POST	/api/project/{projectId}/favorite	프로젝트 좋아요 누르기
GET	/api/project/{projectId}	프로젝트 상세 내용 조회
DELETE	/api/project/{projectId}	프로젝트 삭제
PATCH	/api/project/{projectId}	프로젝트 업데이트
GET	/api/project/{anotherUserId}/page/{pageNumber}	한 유저의 모든 공개 프로젝트 조회
GET	/api/project/search/{pageNumber}	프로젝트 검색
GET	/api/project/page/{pageNumber}	페이지에서 전체 프로젝트 페이징
GET	/api/project/myprojects	현재 로그인한 유저의 모든 프로젝트 조회
GET	/api/project/mypage/{pageNumber}	마이 페이지에서 자신의 프로젝트 페이징
GET	/api/project/libraries/search/{pageNumber}	라이브러리까지 검색 후 페이징
GET	/api/project/dashboard/{pageNumber}	자신이 팔로우하는 유저들(팔로잉)과 자신의 프로젝트 목록 페이지 조회

Table 105: API Specification 03

위 표는 프로젝트와 관련된 로직을 처리하는 API 이다. 프로젝트 CRUD 는 물론, 팔로우, 좋아요, 검색 기능 등이 얽혀 있는 다양한 로직이 존재한다. 또한 프로젝트 및 라이브러리 정보를 불러올 때 데이터베이스 요청의 과부하를 방지하기 위한 페이지네이션이 제공되어 페이지 단위로 불러올 수 있다.

15.2.4 Follow Controller

Type	URI	Description
POST	/api/follow/{followUserId}	모든 공개 프로젝트 조회

GET	/api/follow/followerList	자신을 팔로우하는 유저들(팔로워) 목록 조회
GET	/api/follow/followList	자신이 팔로우하는 유저들(팔로잉) 목록 조회

Table 106: API Specification 04

회원간의 팔로우 및 팔로우 목록 조회를 위한 API 이다.

15.2.5 File Controller

Type	URI	Description
POST	/api/file/uploads	파일 아마존 S3 에 업로드 후 업로드한 정보 반환
DELETE	/api/file/delete	업로드한 파일 삭제

Table 107: API Specification 05

유저의 프로필 이미지 업로드 및 삭제를 위한 API 이다.

15.2.6 Library Count Controller

Type	URI	Description
GET	/api/libraryCount/{topN}	인기 라이브러리 목록 조회
GET	/api/libraryCount/top10	상위 10 개 인기 라이브러리 목록 조회

Table 108: API Specification 06

인기 라이브러리 조회에 사용할 정보를 위한 읽기전용 API 이다.

15.3 DB Schema

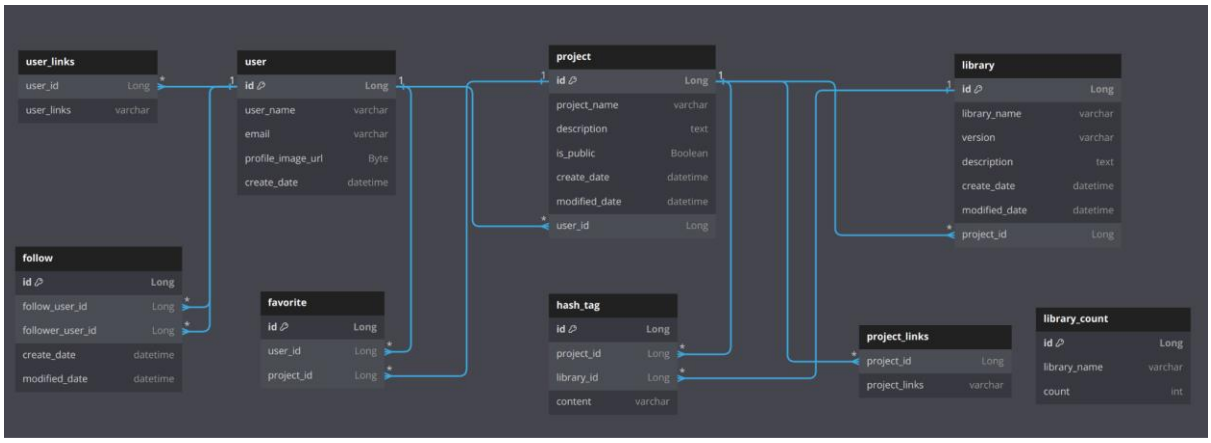


Figure 47: DB Schema

16. Appendix

16.1 User Interface

16.1.1 로그인/회원가입

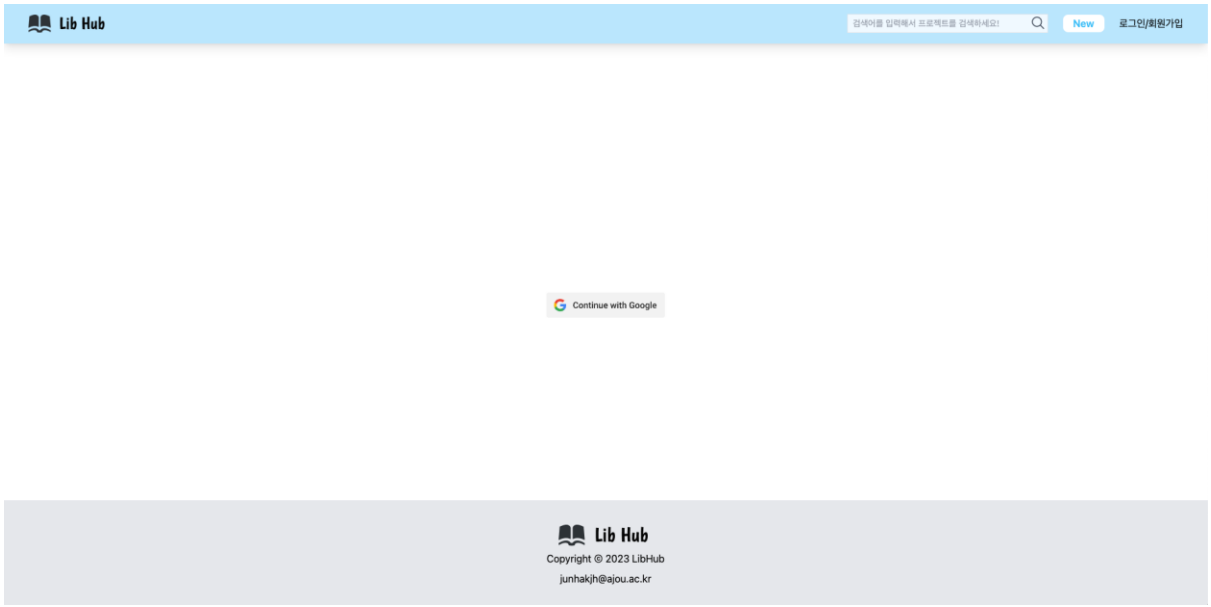


Figure 48: UI - SignIn/SignUp

16.1.2 메인 페이지

Figure 49: UI - Main Page

16.1.3 사이드바

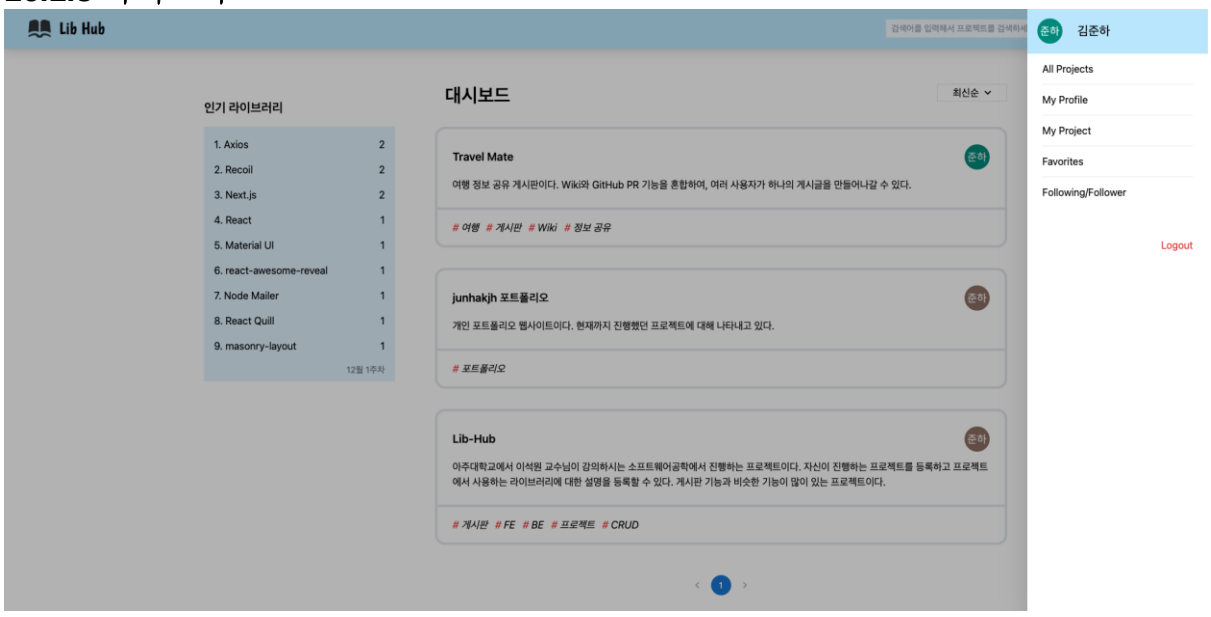


Figure 50: UI - SideBar

16.1.4 프로젝트 등록

Figure 51: UI - New Project

16.1.5 라이브러리 등록

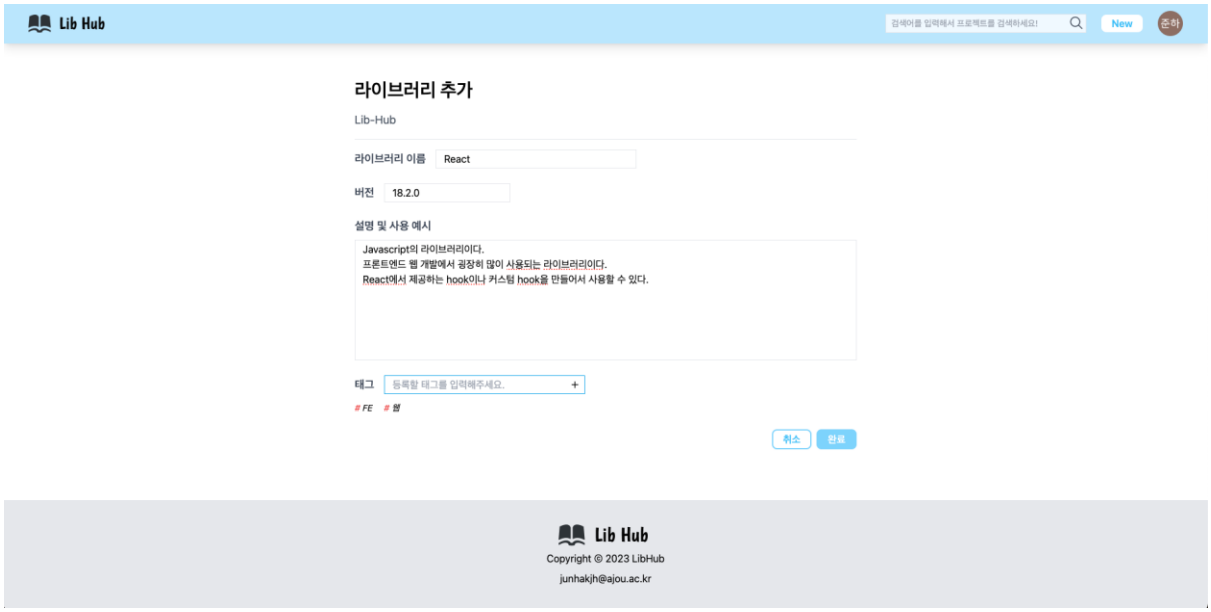


Figure 52: UI - New Library

16.1.6 프로젝트 상세조회

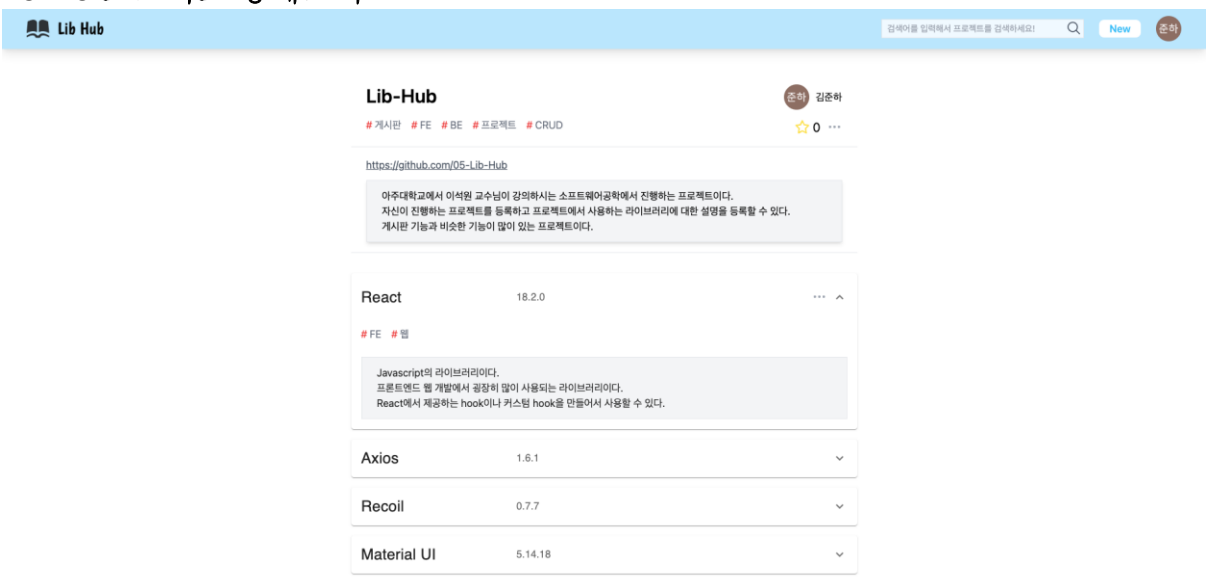


Figure 53: UI - Project Detail

16.1.7 모든 프로젝트

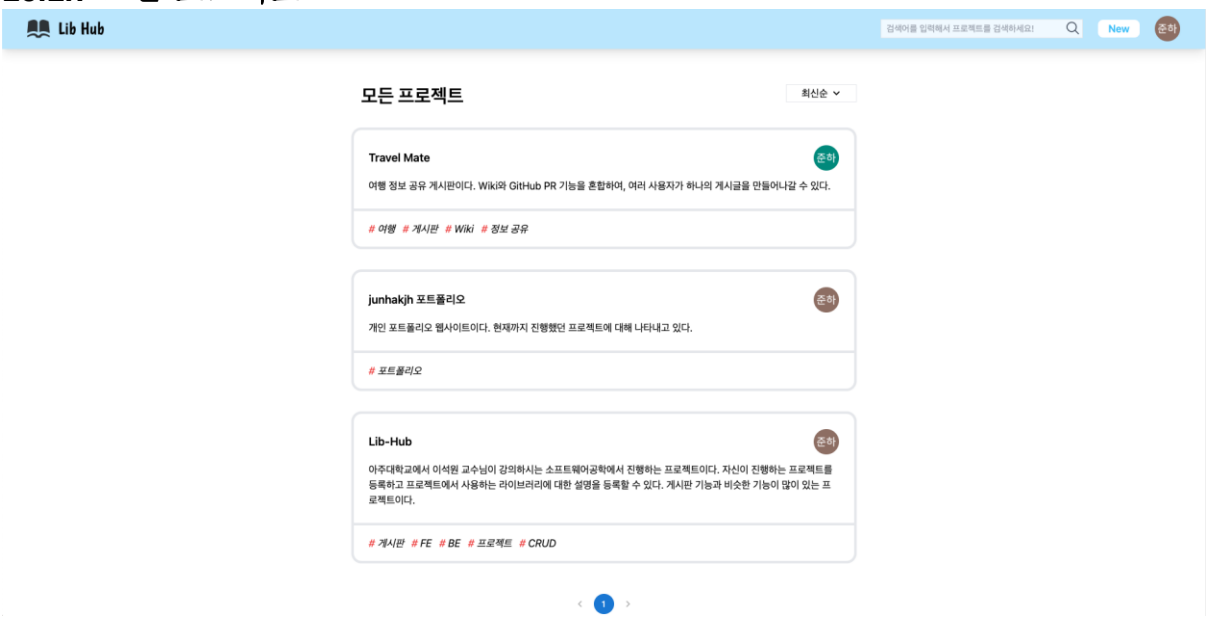


Figure 54: UI - All Projects

16.1.8 프로젝트 검색

Figure 55: UI - Search Projects

16.1.9 내 프로젝트

Figure 56: UI - My Projects

16.1.10 즐겨찾기 프로젝트

Figure 57: UI - Favorite Projects

16.1.11 사용자 프로필 조회

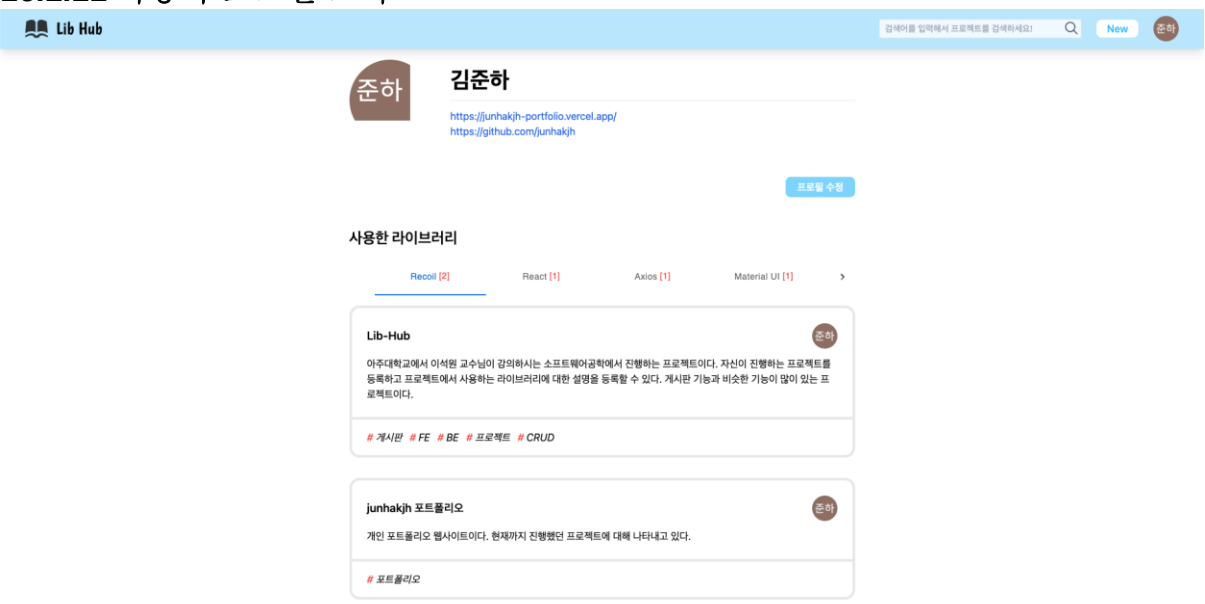


Figure 58: UI - User Profile

16.1.12 프로필 수정

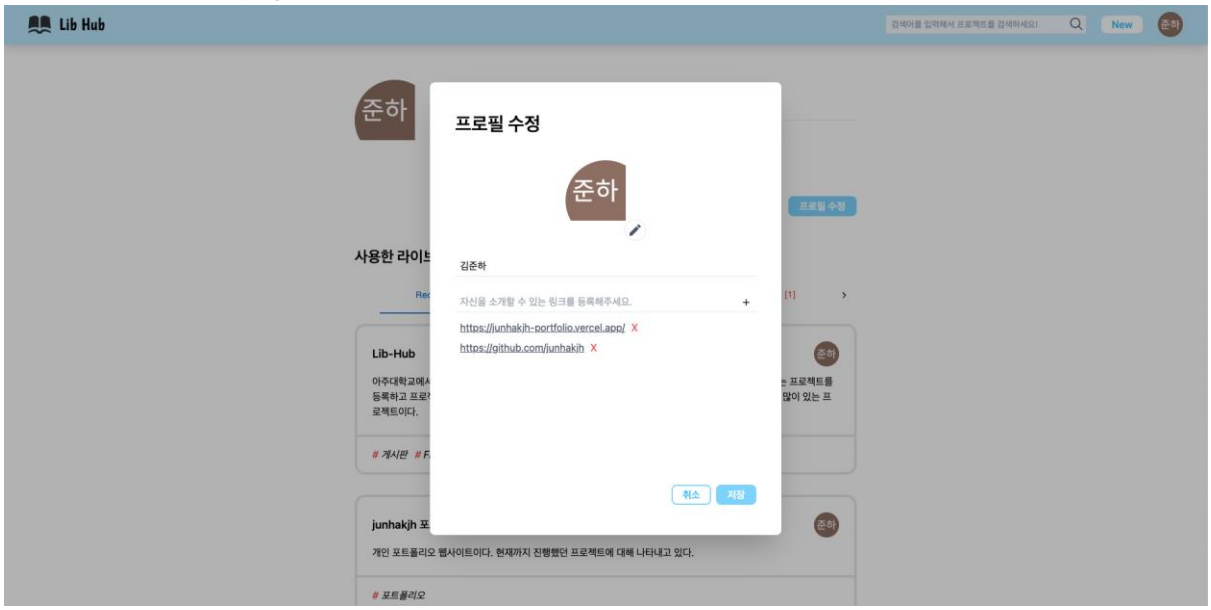


Figure 59: UI - Edit Profile

16.1.13 팔로워/팔로잉

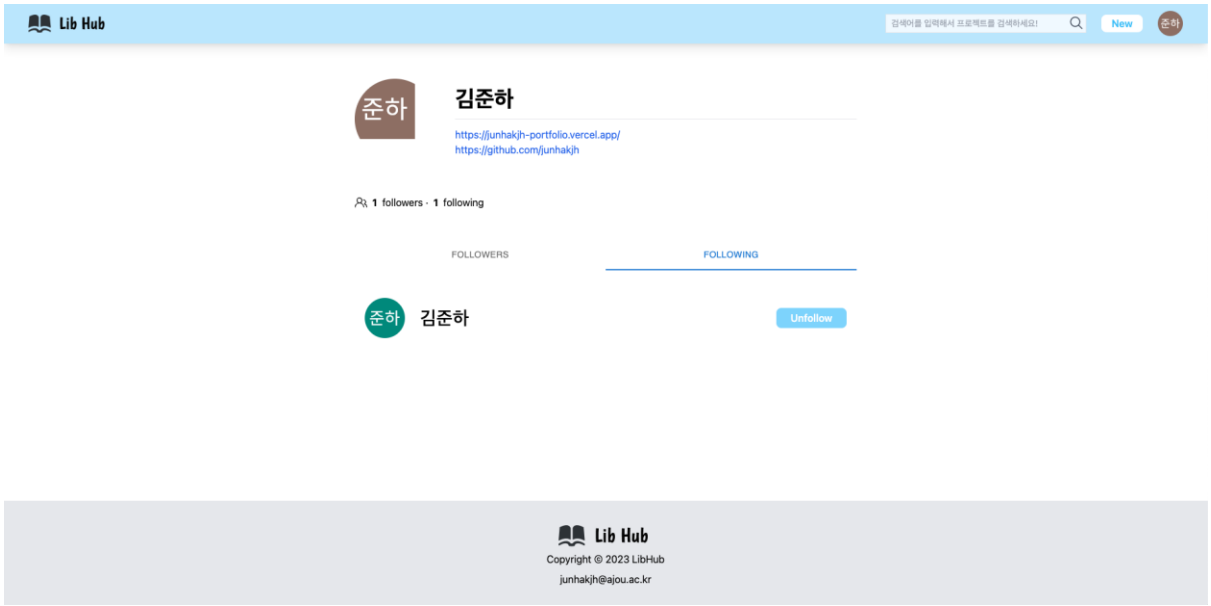


Figure 60: UI - Follower/Following

16.2 Project Status

16.2.1 FE

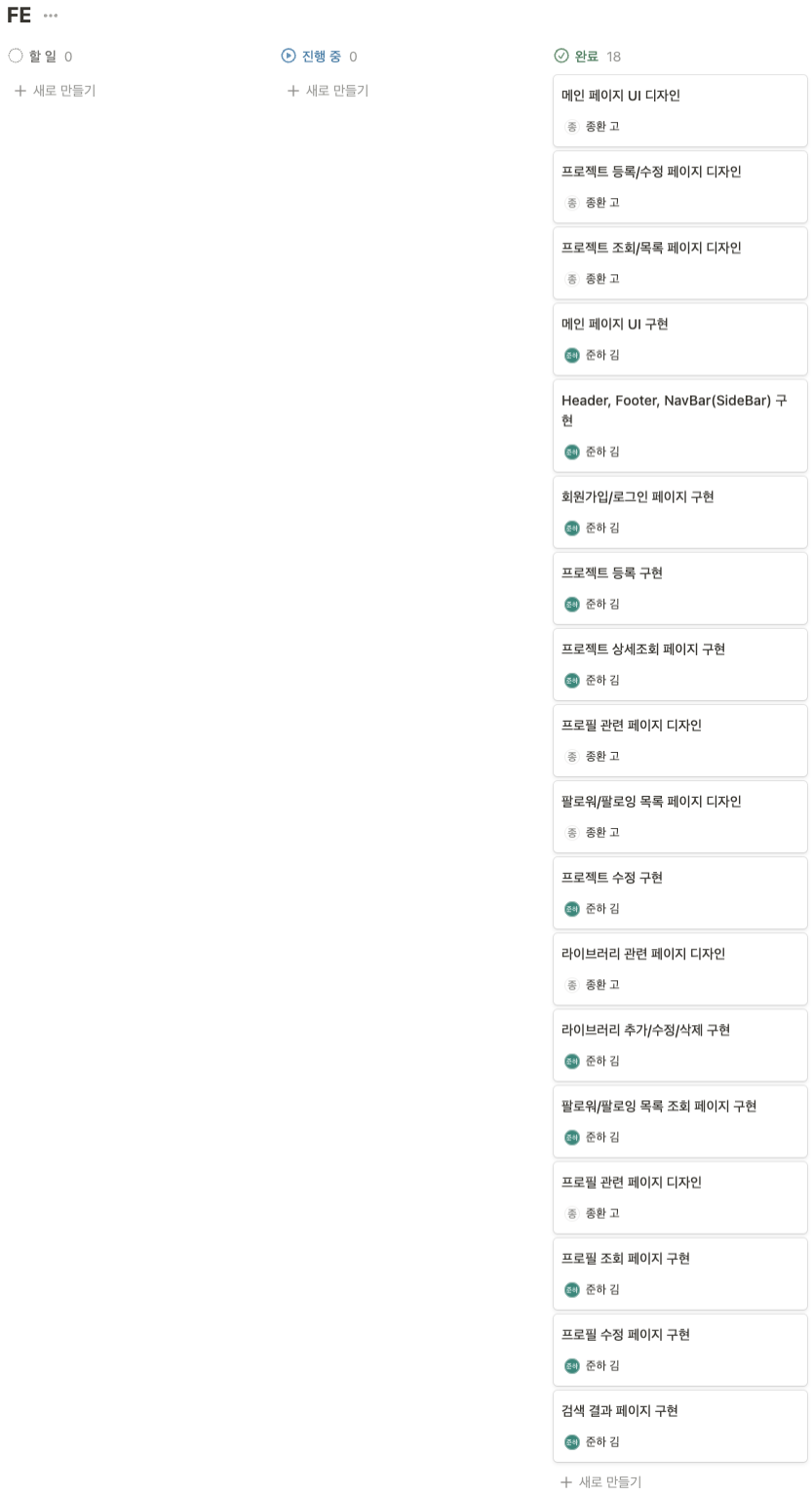


Figure 61: FE - Status

16.2.2 BE

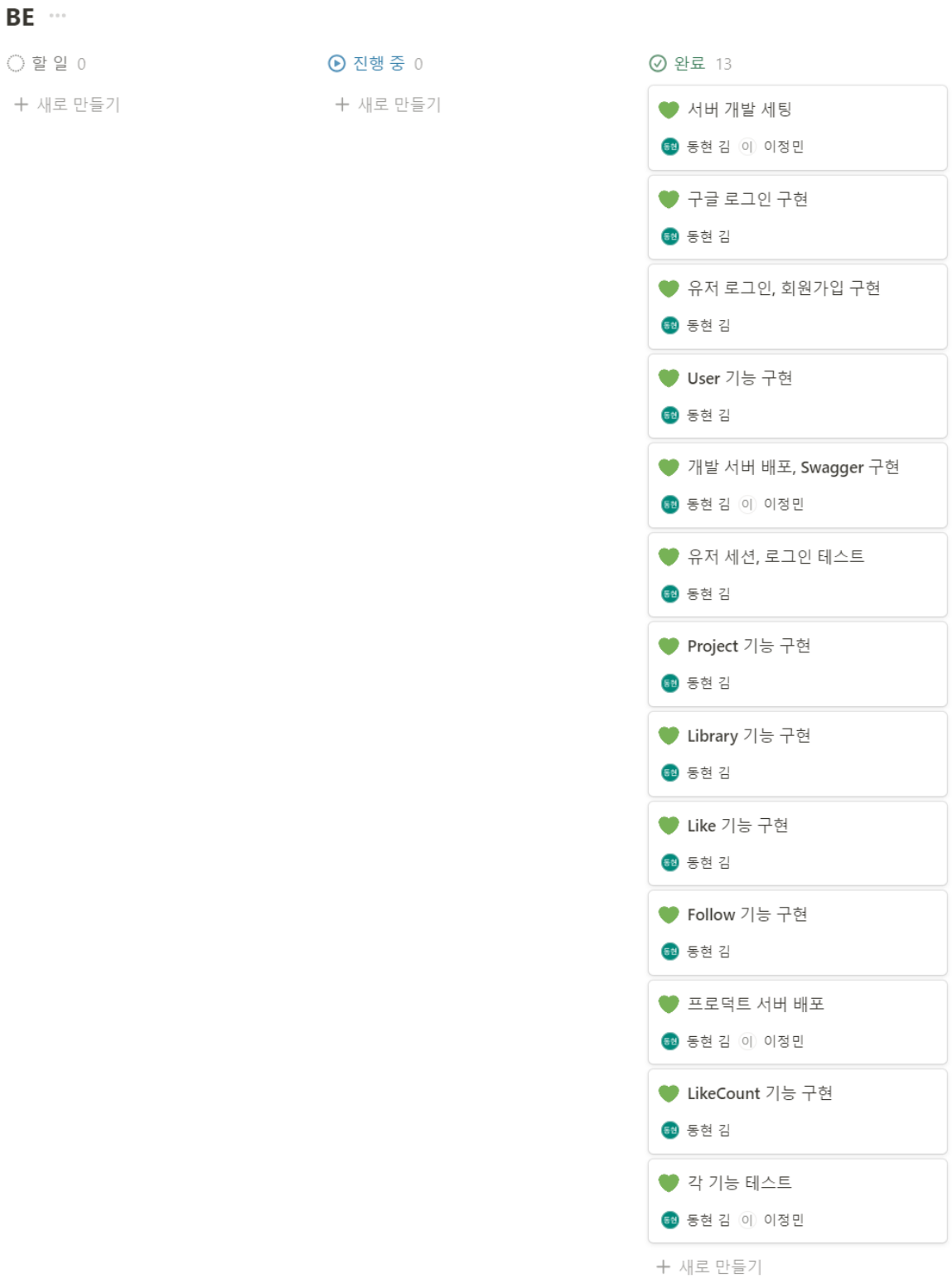


Figure 62: BE - Status

16.3 Members Contribution

고종환 - Use Case Diagram 작성, Sequence Diagram 작성, UI 디자인, testing plan 작성 및 test 진행

김동현 - DB schema 설계, 개발 서버 구축 및 백엔드 개발. 버그 수정 및 테스트 코드 작성. 서버에 프로덕트 배포, Final Report 의 Implementation 내용 추가 및 전체 검수

김준하 - 프론트엔드 개발, Final Report 의 Implementation 작성 및 전체 내용 검수

이정민 - DB schema 설계, Use Case Diagram 작성, Sequence Diagram 전반부 작성, Design Class Diagram 작성, test case 전반부 작성 및 개발 서버 구축

16.4 GitHub

- **05-Lib-Hub -** https://github.com/05-Lib-Hub

17. References

- 권혜미 기자, 기업 10 곳 중 7 곳, 오픈소스 SW 쓴다, 『전자신문 etnews』, 2023-03-26 17:00, https://www.etnews.com/
- DevGoory.(2017. 6. 23.) 유스케이스 다이어그램 - Usecase Diagram. https://googry.tistory.com/2
- Apache JMeter 공식 문서 - https://jmeter.apache.org/usermanual/index.html
- Github 홈페이지 - https://github.com/