

# 이슈관리 시스템 개발 보고서

8조

1. 프로젝트 내용 요약
  - a. 프로젝트 개요
  - b. 구현된 **MUST** 기능
2. 프로젝트 요구 정의 및 분석
  - a. 유스케이스 다이어그램
  - b. 유스케이스 명세
  - c. 도메인 모델
  - d. **System-sequence** 다이어그램
3. 설계
  - a. 클래스 다이어그램
  - b. 시퀀스 다이어그램
  - c. **ER** 다이어그램
  - d. **GRASP** 패턴 원칙의 적용
4. 주요 기능 구현
5. JUnit 테스트 수행 내역
  - a. 테스트 케이스
  - b. 테스트 결과
6. GitHub 프로젝트 활용
  - a. GitHub 프로젝트 주소
  - b. **Progress History**
  - c. 팀원별 기여

## 1. 프로젝트 내용 요약

### a. 프로젝트 개요

소프트웨어 개발 과정에서 이슈 관리 시스템은 필수적인 도구로 자리 잡고 있다. 이슈 관리 시스템은 조직이나 단체가 개발 중에 발생하는 다양한 문제점이나 작업 항목을 체계적으로 관리할 수 있도록 도와준다. 이러한 시스템은 종종 트러블 티켓 시스템(trouble ticket system)이라고 불리며, JIRA, TRAC, Bugzilla, Backlog 등 다양한 상용 및 오픈소스 소프트웨어가 존재한다.

본 프로젝트에서는 이슈 등록과 관리 기능을 중심으로 하는 이슈 관리 시스템을 설계하고 구현하는 것을 목표로 한다. 객체지향 분석과 설계 개념을 적극적으로 활용하여 시스템을 구축하며, 각 설계 선택의 이유를 문서화하여 설계 과정의 투명성과 이해도를 높이하고자 한다. 특히 기능 확장성과 설계 개선 활동이 용이하도록 JUnit을 이용한 테스트케이스를 작성하고, 이를 통해 시스템의 신뢰성을 보장하도록 하게 한다.

시스템에 저장된 데이터는 영속적인 저장소에 보관하여 시스템 재시작 시에도 기존 데이터를 지속적으로 사용할 수 있도록 한다. 이를 위해 DBMS 또는 파일 시스템을 활용할 수 있으며, 데이터의 영속성을 보장하는 방식으로 설계한다.

또한, 본 시스템은 GUI와 웹 인터페이스를 통해 동작하도록 구현한다. 설계 및 구현 과정에서 사용자 인터페이스와 응용 로직을 분리하여, UI가 변경되더라도 로직 이하의 계층은 수정 없이 재사용할 수 있도록 한다. 이러한 설계 원칙을 준수함으로써 시스템의 유지보수성과 확장성을 확보하고, 프로젝트에 이를 명확히 나타내어 후속 작업의 용이성을 높이하고자 한다.

결론적으로, 본 프로젝트는 조직의 이슈 관리 요구사항을 충족하는 맞춤형 시스템을 제공함으로써 개발 프로세스의 효율성을 향상시키고, 향후 확장 및 개선이 용이한 구조를 갖춘 이슈 관리 시스템의 모범 사례를 제시하는 것을 목표로 한다.

### b. 구현된 MUST 기능

본 프로젝트에서 구현된 MUST 기능들은 다음과 같다.

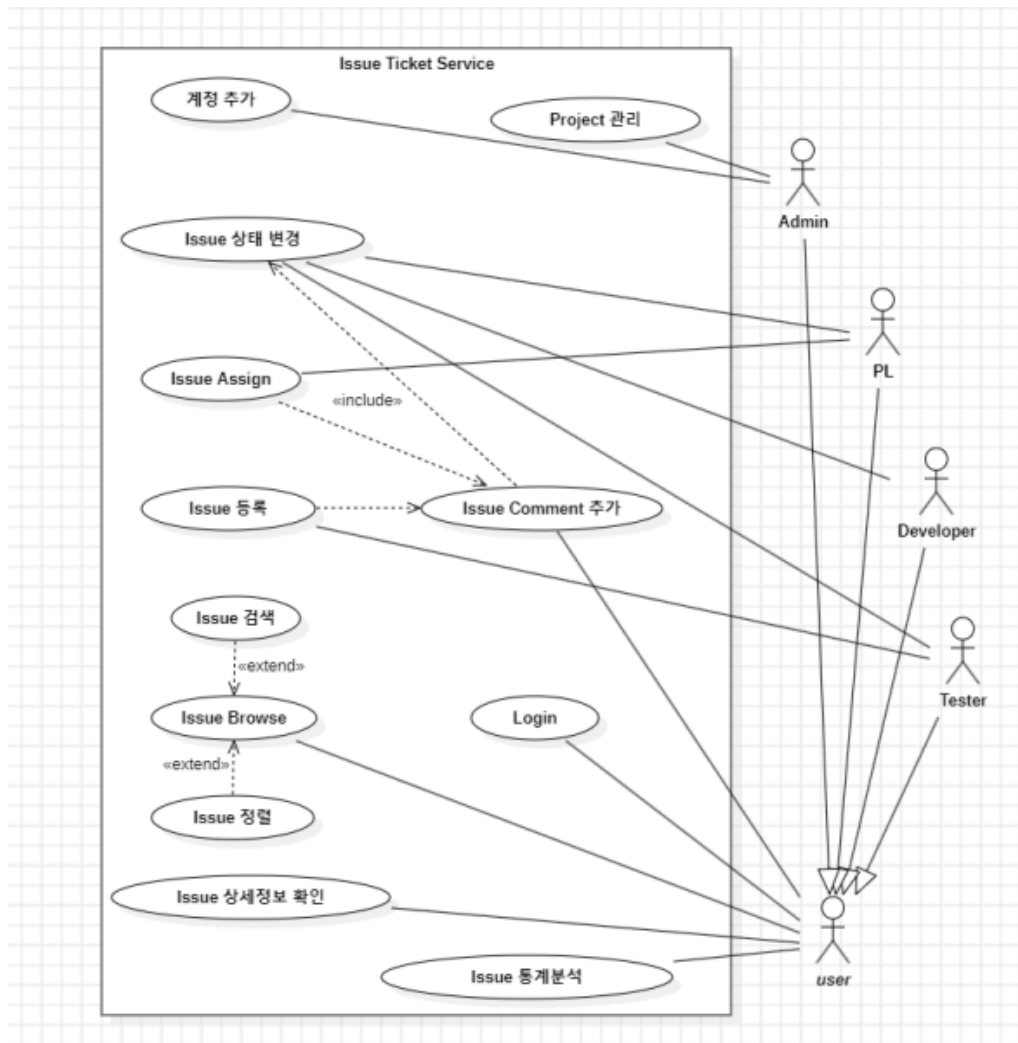
- 계정 추가: 계정 추가는 Admin 권한을 가진 사용자가 추가하며, 사용자의 이름, ID, 비밀번호, 권한을 입력함으로써 추가된다.
- 이슈 브라우즈 및 검색: 이슈 검색은 특정 프로젝트의 이슈창에서 수행하며, 제목(Title), 우선순위(Priority), 상태(State), 유형(Type), 이슈 발견자(Reporter), 이슈 해결자(Assignee) 중 하나를 선택하고 이를 String 형식으로 입력하여 검색할 수 있다.
- 이슈 등록: 이슈 등록은 사용자가 이슈의 제목, 우선순위, 상태, 유형, 내용(Description)을 입력함으로써 추가하며, 이때 이슈를 추가하는 사용자는 자동으로 기록된다.
- 이슈 코멘트 추가: 이슈 코멘트 추가는 등록되어 있는 이슈를 선택하고 내용을 입력함으로써 추가된다.
- 이슈 상세 정보 확인: 이슈 상세 정보는 프로젝트 내의 이슈를 선택하여 확인할 수 있다.
- 이슈 상태 변경: 이슈 상태 변경은 이슈 상세 정보에서 변경할 수 있으며, 이때 이슈 해결자의 배정 및 우선순위/상태를 변경할 수 있다.

- 이슈 통계 분석: 이슈 통계 분석에서 사용자는 이슈의 월간 생성 횟수와 이슈의 우선순위, 상태, 유형 비율을 막대 그래프를 통해 확인할 수 있다.
- **assignee** 자동 추천 기능: 이슈 등록 후 해결자를 할당하기 전 추천 기능은 현 이슈와 이전 이슈들의 내용/코멘트의 문자열 유사도값을 측정하여 가장 유사도가 높은 개발자를 추천해준다.

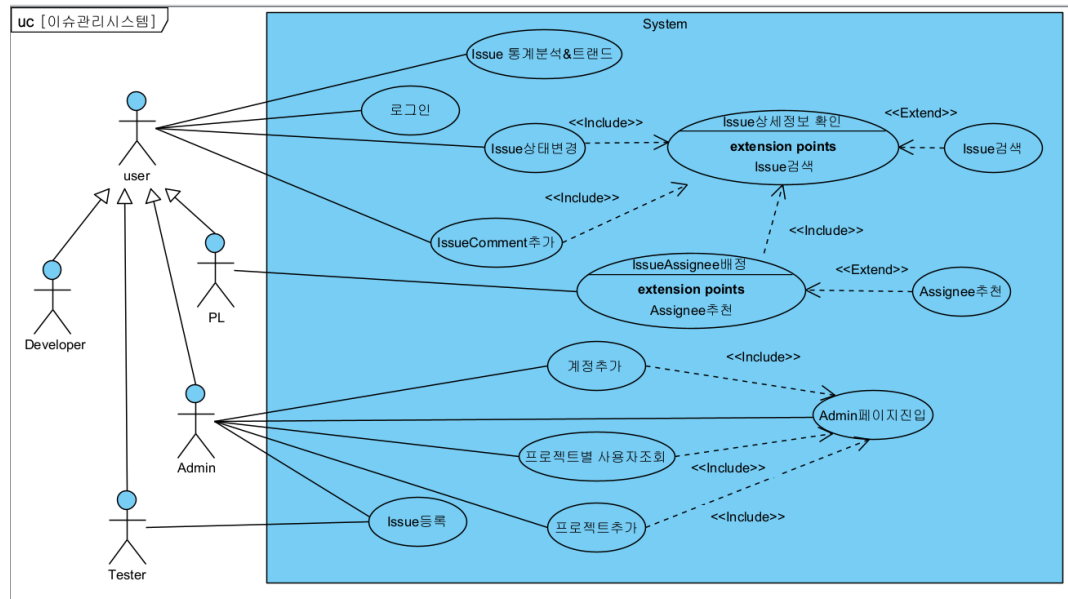
## 2. 프로젝트 요구 정의 및 분석

### a. 유스케이스 다이어그램

#### i. 초기 유스케이스 다이어그램



#### ii. 최종 유스케이스 다이어그램



b. 유스케이스 명세

아래의 유스케이스 명세는 **MUST** 기능을 포함한 **13개의 유스케이스 명세**를 서술하고 있다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	로그인
액터	사용자
시작 조건	사용자가 이슈 관리 시스템에 로그인 되어있지 않아야 함.
기본 흐름	1. 사용자는 ID와 비밀번호를 입력한다. 2. 시스템은 입력된 정보를 DB에 저장된 사용자 정보와 대조한다. 3. 입력된 정보가 일치할 시 사용자는 로그인에 성공한다.
대안 흐름	대안 흐름 A1 : 로그인 실패 기본 흐름 2에서 분기 A1.1. 입력된 정보가 불일치할 경우 시스템이 오류 메시지를 표시한다. A1.2. 기본흐름 1로 이동한다.
종료 조건	사용자는 이슈 관리 시스템에 로그인된 후, 프로젝트 목록 페이지에 위치한다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	Admin페이지 진입
액터	사용자 - Admin
시작 조건	Admin이 이슈 관리 시스템에 로그인 되어야 함.

기본 흐름	1. 사용자가 <b>Admin</b> 버튼을 누른다. 2. 시스템이 <b>Admin</b> 페이지를 <b>load</b> 한다.
대안 흐름	대안 흐름 <b>A1</b> : 사용자 권한 미달 기본 흐름 <b>1</b> 에서 분기 <b>A1.1.</b> 사용자의 권한이 <b>Admin</b> 이 아닐 경우 접근 제한 메시지를 출력한다.
종료 조건	사용자가 <b>Admin</b> 페이지에 위치해있다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	프로젝트 추가
액터	사용자 - Admin
시작 조건	<b>Admin</b> 이 이슈 관리 시스템에 로그인 되어야 함.
기본 흐름	<u>1.include(Admin페이지 진입)</u> 2. 사용자가 “ <b>Add Project</b> ”를 선택한다. 3. 사용자가 사용자 정보인 제목, 내용, <b>PL</b> , <b>Dev</b> , <b>Tester</b> 를 입력한다. 4. 사용자가 “ <b>Add</b> ” 버튼을 누른다. 5. 시스템이 입력된 정보를 기반으로 한 프로젝트를 추가하고 저장한다.
대안 흐름	대안 흐름 <b>A1</b> : 비어있는 입력 오류 기본 흐름 <b>4</b> 에서 분기 <b>A1.1.</b> 사용자가 프로젝트 제목과 내용을 전부 기입하지 않은 경우 시스템이 오류 메시지를 표시한다. <b>A1.2.</b> 사용자가 제목 및 내용 정보를 수정한다. <b>A1.3.</b> 기본 흐름 <b>4</b> 로 이동한다.  대안 흐름 <b>A2</b> : 잘못된 사용자 수 오류 기본 흐름 <b>4</b> 에서 분기 <b>A2.1.</b> 사용자가 시스템에서 요구하는 프로젝트 인원 수만큼 선택하지 않은 경우 시스템이 오류 메시지를 표시한다. <b>A2.2.</b> 사용자가 인원 선택 정보를 수정한다. <b>A2.3.</b> 기본 흐름 <b>4</b> 로 이동한다.
종료 조건	새로운 프로젝트가 시스템에 추가된다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	계정 추가
액터	사용자 - Admin
시작 조건	<b>Admin</b> 이 이슈 관리 시스템에 로그인 되어야 함.
기본 흐름	<u>1. include(Admin페이지 진입)</u> 2. 사용자가 “ <b>Add User</b> ”를 선택한다.

	<p>3. 사용자가 사용자 정보인 이름, ID, 비밀번호, 레벨을 입력한다.</p> <p>4. 사용자가 “Add” 버튼을 누른다.</p> <p>5. 시스템이 입력된 정보를 기반으로 한 계정을 추가하고 저장한다.</p>
대안 흐름	<p>대안 흐름 A1 : 비어있는 입력 오류 기본 흐름4에서 분기</p> <p>A1.1. 사용자가 기입해야하는 정보를 전부 작성하지 않은 경우 시스템이 오류 메시지를 표시한다.</p> <p>A1.2. 사용자가 정보를 수정한다.</p> <p>A1.3. 기본 흐름4로 이동한다.</p>
종료 조건	새로운 계정이 시스템에 추가된다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	프로젝트 별 사용자 조회
액터	사용자 - Admin
시작 조건	사용자가 이슈 관리 시스템에 로그인 되어야 함.
기본 흐름	<p>1.include(Admin페이지 진입)</p> <p>2. 사용자가 “Project Users”를 선택한다.</p> <p>3. 시스템은 지금까지 생성된 사용자들의 정보를 프로젝트 별로 분류한다.</p> <p>4. 사용자는 보고자하는 프로젝트를 드롭다운 메뉴에서 선택하고 적용버튼을 누른다.</p> <p>5. 시스템은 해당 프로젝트의 사용자 목록을 표시한다.</p>
대안 흐름	<p>대안 흐름 A1 : 프로젝트 없음 기본 흐름2에서 분기</p> <p>A1.1. 확인할 프로젝트나 사용자가 없는 경우 시스템은 드롭다운 메뉴를 비워둔 채로 둔다.</p>
종료 조건	Admin은 사용자들의 목록을 확인할 수 있다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	이슈 상세정보 확인
액터	사용자
시작 조건	사용자가 이슈 관리 시스템에 로그인 되어야 함. 확인할 수 있는 이슈가 사용자가 소속된 프로젝트 내에 존재해야 함.
기본 흐름	<p>1. 사용자가 프로젝트 목록 페이지에 진입한다.</p> <p>2. 시스템이 사용자가 소속된 프로젝트 목록을 나열한다.</p> <p>3. 사용자는 확인하고자 하는 이슈가 있는 프로젝트를 선택한다.</p>

	<p><u>확장점: 이슈 검색</u></p> <p>4. 사용자가 이슈 목록에서 확인할 이슈를 선택한다.</p> <p>5. 시스템은 선택된 이슈의 상세 정보를 표시한다.</p>
대안 흐름	<p>대안 흐름 A1 : Admin Case</p> <p>기본 흐름 1에서 분기</p> <p>A1.1. 시스템이 모든 프로젝트 목록을 나열한다.</p> <p>A1.2. 기본 흐름3으로 이동한다.</p>
종료 조건	사용자는 이슈의 상세 정보를 확인할 수 있다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	이슈 검색
액터	사용자
시작 조건	사용자가 이슈 관리 시스템에 로그인 되어야 함. 사용자가 이슈 목록 페이지에 위치해야함.
기본 흐름	<p>1. 사용자는 이슈 검색 옵션을 드롭다운 메뉴에서 선택한다.</p> <p>2. 사용자가 검색하고자 하는 텍스트를 검색창에 입력한다.</p> <p>3. 사용자가 검색버튼을 누른다.</p> <p>4. 시스템은 모든 이슈 중에 선택된 옵션에 입력된 텍스트가 포함된 이슈들을 찾는다.</p> <p>5. 찾아낸 이슈들을 목록에 출력한다.</p>
대안 흐름	<p>대안 흐름 A1 : 검색결과 없음</p> <p>기본 흐름4에서 분기</p> <p>A1.1. 검색 결과가 없는 경우 시스템은 빈 이슈 목록을 출력한다.</p>
종료 조건	사용자는 자신이 설정한 검색 결과에 해당하는 이슈 목록을 확인할 수 있다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	이슈 등록
액터	사용자-Tester, Admin
시작 조건	Tester 또는 Admin이 이슈 관리 시스템에 로그인 되어야 함.
기본 흐름	<p>1. 사용자는 프로젝트를 선택한 후 “Issue Add”를 선택한다.</p> <p>2. 사용자가 이슈에 대한 상세 정보를 입력한다.</p> <p>3. 사용자는 Add버튼을 클릭한다.</p> <p>4. 시스템은 입력된 정보를 바탕으로 새로운 이슈를 등록한다.</p>

	5. 시스템은 등록 완료 메시지를 사용자에게 표시한다.
대안 흐름	<p>대안 흐름 A1 : 사용자 권한 불만족 기본 흐름1에서 분기 A1.1.사용자가 <b>Tester</b>또는 <b>Admin</b>이 아닐 경우 시스템은 [이슈 등록 창 입장 불가] 오류 메시지를 출력한다.</p> <p>대안 흐름 A2 : 비어있는 입력 오류 기본 흐름3에서 분기 A2.1. 필수 정보가 누락된 경우 시스템은 오류 메시지를 표시한다. A2.2. 기본 흐름2로 이동한다.</p>
종료 조건	새로운 이슈가 시스템에 등록된다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	이슈 코멘트 추가
액터	사용자
시작 조건	사용자가 이슈 관리 시스템에 로그인 되어야 함. 추가할 코멘트가 있는 이슈가 존재해야 함.
기본 흐름	<p>1.include(이슈 상세정보 확인) 2. 사용자가 “<b>New comment</b>”를 선택한다. 3. 시스템이 <b>AddComment</b>모달을 띄운다. 4. 사용자가 코멘트를 입력한다. 5. 사용자가 <b>Add</b>버튼을 누른다. 6. 시스템은 코멘트를 해당 이슈에 날짜, 사용자 정보와 함께 추가한다. 6. 시스템은 추가 완료 메시지를 사용자에게 표시한다.</p>
대안 흐름	<p>대안 흐름 A1 : 비어있는 입력 오류 기본 흐름5에서 분기 A1.1. 코멘트 입력란이 비어 있는 경우 시스템은 오류 메시지를 출력한다. A1.2. 기본 흐름3으로 이동한다.</p> <p>대안 흐름 A2 : 닫힌 이슈 기본 흐름 2에서 분기 A2.1. 시스템은 상태가 <b>Closed</b>인 이슈에 사용자가 댓글을 추가하려고 시도할경우 경고메시지를 출력한다.</p>
종료 조건	이슈에 코멘트가 추가된다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	이슈 상태 변경
액터	사용자



시작 조건	사용자가 이슈 관리 시스템에 로그인 되어야 함. 상태를 변경할 수 있는 이슈가 존재해야 함.
기본 흐름	<p><u>1.include(이슈 상세정보 확인)</u>  2.사용자가 <b>issue</b>의 상태를 변경한 후 <b>Apply</b> 버튼을 누른다.  3. 시스템은 이슈의 상태를 업데이트한다.  4. 시스템은 상태 변경 완료 메시지를 사용자에게 표시한다.</p>
대안 흐름	<p>대안 흐름 A1 : <b>Closed</b> 상태에서 상태 변경  기본 흐름1에서 분기  A1.1. <b>issue</b>의 상태가 <b>Closed</b>인 경우, 시스템은 상태 변경 드롭다운 메뉴들을 <b>Status</b> 제외하고 모두 비활성화한다.</p> <p>대안 흐름 A2 : <b>Closed</b>에서 다시 정상 상태로 복구  대안 흐름 <b>A1.1</b>에서 분기  A2.2. <b>issue</b>의 상태를 <b>status</b> 드롭다운 메뉴에서 <b>Reopened</b>로 설정한 후 <b>Apply</b> 버튼을 누른다.  A2.3. 시스템은 이슈의 상태를 <b>Reopened</b>로 업데이트 하면서 모든 상태 메뉴의 비활성화를 해제한다.  A2.4. 기본 흐름2로 이동한다.</p> <p>대안 흐름 A3 : <b>New</b> 상태로 역행  기본 흐름2에서 분기  A3.3. 이미 <b>Assignee</b>가 할당된 상태에서 사용자가 다시 <b>New</b>로 상태를 되돌리려고 시도할 경우 [상태 역행 불가] 에러 메시지를 출력한다.</p> <p>대안 흐름 A4 : <b>Fixer</b>지정  기본 흐름 3에서 분기  A4.1. 이슈가 <b>Resolved</b>이상의 상태로 변경된 경우, 시스템은 <b>Fixer</b> 칸에 현재 <b>Assignee</b>의 이름을 출력한다.  A4.2. 기본 흐름4로 이동한다.</p> <p>대안 흐름 A5 : <b>Fixer</b>를 모르는 상황  기본 흐름 3에서 분기  A5.1. 이슈가 <b>Resolved</b> 상태 미만일 경우 아직 <b>Fixer</b>를 특정할 수 없으므로 시스템은 <b>Fixer</b> 칸에 <b>N/A</b>를 출력한다.  A5.2. 기본 흐름4로 이동한다.</p> <p>대안 흐름 A6 : <b>Fixer</b>가 지정된 상황에서 <b>Assignee</b>변경  기본 흐름 2에서 분기  A6.1. <b>Fixer</b>가 이미 정해진 상황에서 사용자가 해당 이슈의 <b>Assignee</b>를 변경하려고 했을 경우, 시스템은 에러메시지를 출력한다.</p>
종료 조건	이슈의 상태가 변경된다.

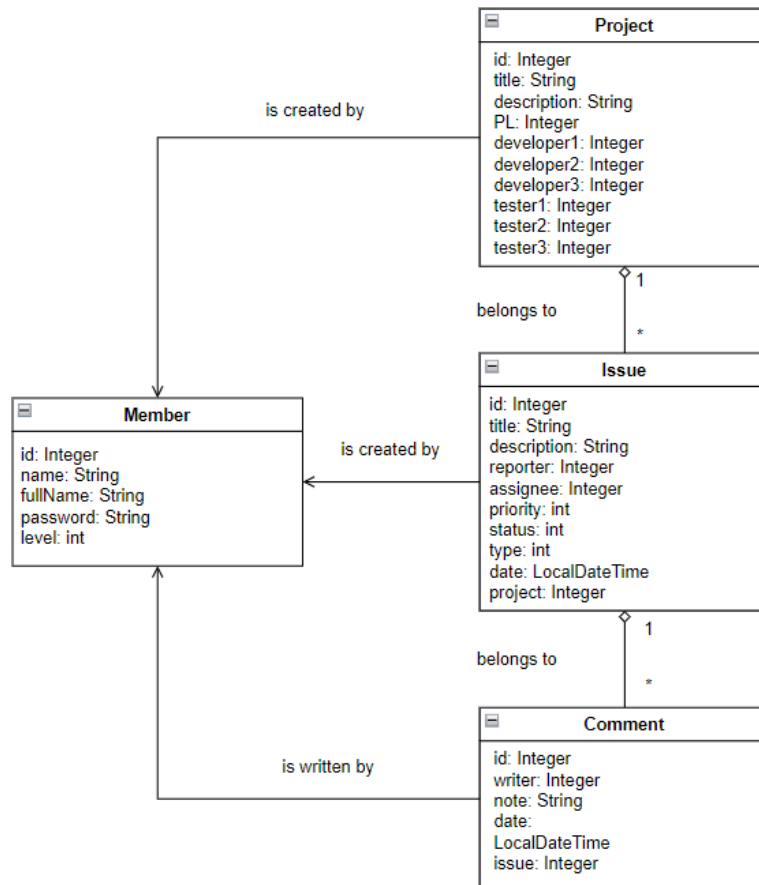
시스템 제목	이슈 관리 시스템
유스케이스 이름	Assignee추천
액터	사용자
시작 조건	사용자가 이슈 관리 시스템에 로그인 되어야 함.
기본 흐름	1. 사용자가 이슈 상태 변경 페이지에 진입한다. 2. 시스템이 존재하는 모든 <b>issue</b> 정보를 기반으로 현재 <b>issue</b> 에 가장 적합한 <b>Assignee</b> 를 자동으로 찾아낸다. 3. 시스템이 이슈 상태 변경 페이지의 <b>Assignee selection</b> 옆에 추천 <b>Assignee</b> 의 <b>name</b> 을 출력한다.
대안 흐름	대안 흐름 A1 : 알고리즘 오류 기본 흐름 2에서 분기 A1.1. 추천 알고리즘에 사용할 <b>issue</b> 데이터가 없을 경우, 추천 <b>Assignee</b> 의 <b>name</b> 으로 “N/A”를 반환한다. A1.2. 기본 흐름3으로 이동한다.
종료 조건	이슈상태변경페이지의 <b>Assignee</b> 의 <b>selection</b> 이 시스템이 추천한 <b>Assignee</b> 의 <b>name</b> 과 일치한다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	이슈 <b>Assignee</b> 배정
액터	사용자 - PL, Admin
시작 조건	사용자가 이슈 관리 시스템에 로그인 되어야 함. <b>issue</b> 의 <b>status</b> 가 <b>Closed</b> 이면 안됨.
기본 흐름	<u>1.include(이슈 상세정보 확인)</u> <u>확장점: Assignee추천</u> 2. 사용자(PL)는 이슈에 <b>Assign</b> 할 <b>Developer</b> 를 선택한 후 <b>Apply</b> 버튼을 누른다. 3. 시스템은 해당 <b>issue</b> 에 선택된 <b>developer</b> 를 <b>assignee</b> 로 지정하며, <b>issue</b> 의 <b>status</b> 가 <b>New</b> 였다면 <b>Assigned</b> 로 변경한다. 4. 시스템은 상태 변경 완료 메시지를 사용자에게 표시한다.
대안 흐름	대안 흐름 A1 : PL 권한 미달 기본 흐름1에서 분기 A1.1. 사용자의 권한이 <b>PL</b> 미만일 경우 시스템이 <b>Assignee</b> 할당을 위한 <b>Developer</b> 드롭다운 메뉴가 비활성화시킨다.  대안 흐름 A2 : <b>New</b> 가 아닌 상태에서 <b>Assignee</b> 변경

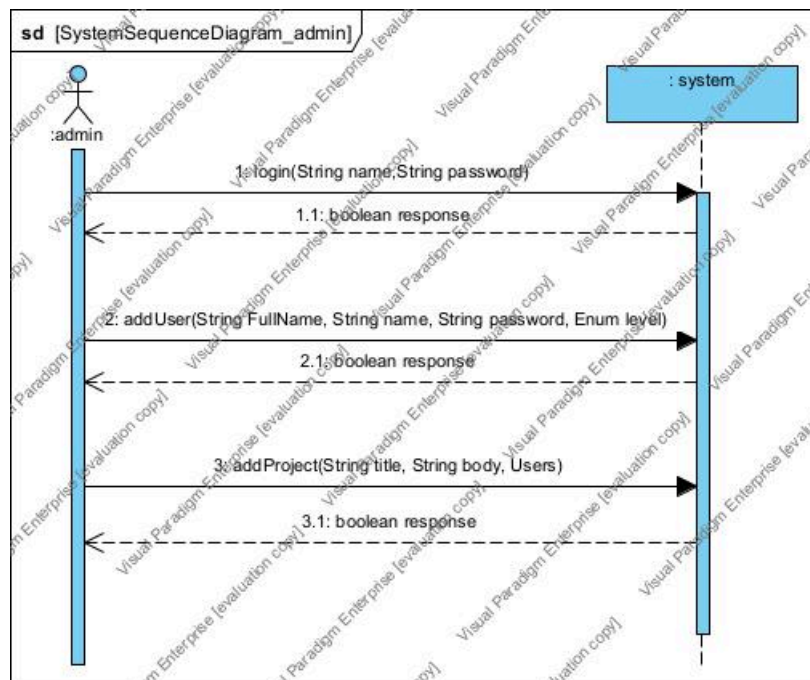
	<p>기본 흐름2에서 분기</p> <p>A2.1. 시스템은 해당 issue에 선택된 developer를 assignee로 지정한다.</p> <p>A2.2. 기본 흐름4로 이동한다.</p> <p>대안 흐름 A3 : Closed 상태에서 Assignee 변경</p> <p>기본 흐름1에서 분기</p> <p>A3.1. issue의 상태가 Closed인 경우, 시스템은 Assignee 드롭다운 메뉴를 비활성화한다.</p>
종료 조건	해당 이슈에서 출력되는 Assignee의 name이 변경 전과 다르다.

시스템 제목	이슈 관리 시스템
유스케이스 이름	이슈 통계 분석&트렌드
액터	사용자
시작 조건	사용자가 이슈 관리 시스템에 로그인 되어야 함.
기본 흐름	<p>1. 사용자가 “Statistics”를 선택한다.</p> <p>2. 시스템은 지금까지 생성된 이슈들의 정보에 따라 이슈 통계 및 트렌드를 분석한다.</p> <p>3. 시스템은 분석 결과를 사용자에게 표시한다.</p>
대안 흐름	<p>대안 흐름 A1 : 데이터 부족</p> <p>기본 흐름 2에서 분기</p> <p>A1.1. 분석할 데이터가 부족한 경우 시스템은 빈 분석 결과를 표시한다.</p>
종료 조건	사용자는 이슈 통계 분석 결과를 확인한다.

### c. 도메인 모델



d. System-sequence 다이어그램  
i. admin

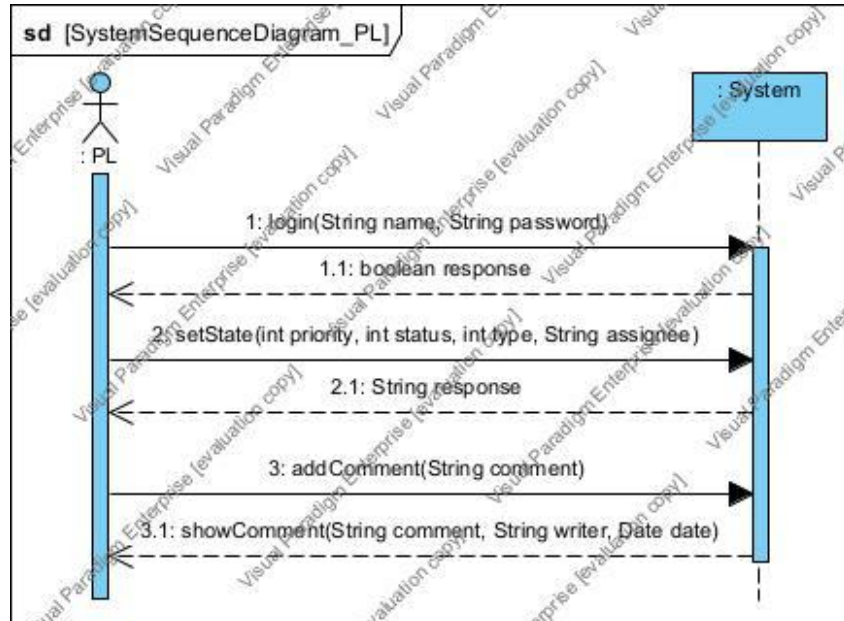


1. admin은 시스템에 id와 password를 입력하고 system은 이것을 name과 password로 받아와 해당 계정이 있으면 로그인 후 home으로 이동

해당 계정이 없으면 시스템이 로그인 실패를 알린다.

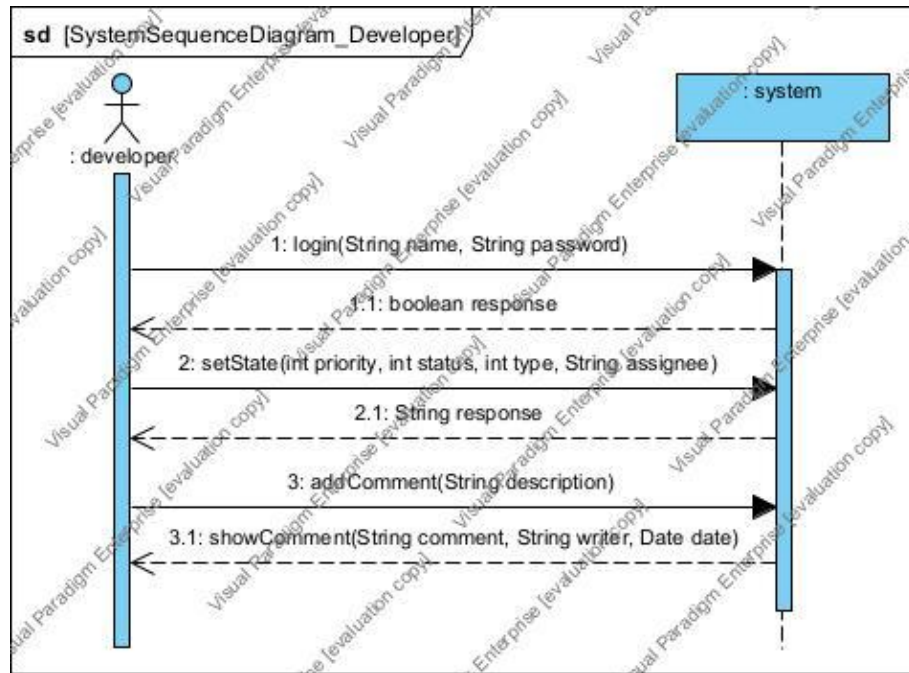
2. **admin**만 사용자 계정을 추가할 수 있으며 이름 **ID Password**를 입력하고 **Level**을 선택하고 **add**로 계정이 추가된다. **ID**가 기존에 있는 경우 시스템이 중복된 아이디임을 알린다.
3. **admin**만 프로젝트를 추가할 수 있으며 프로젝트 제목과 내용을 채워넣고 **PL 1명, Dev 3명, Tester 3명**을 프로젝트에 추가한다. 프로젝트가 성공적으로 생성되면 시스템이 성공을 알린다.

ii. pl



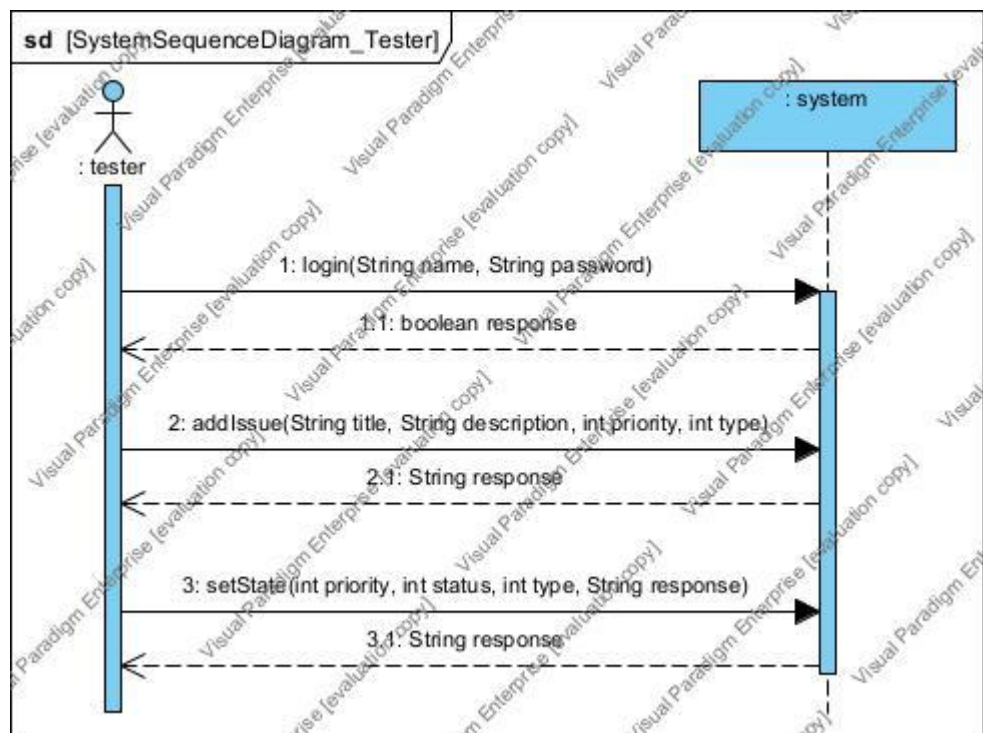
1. PL은 시스템에 **id**와 **password**를 입력하고 **system**은 이것을 **name**과 **password**로 받아와 해당 계정이 있으면 로그인 후 **home**으로 이동 해당 계정이 없으면 시스템이 로그인 실패를 알린다.
2. PL은 이슈의 상태를 변경할 수 있으며 **priority**와 **status**를 고를 수 있다. 시스템이 제공하는 **Recommend Assignee**를 참고하여 **assignee**를 선택할 수 있다.
3. PL은 **Issue**에 **Comment**를 추가할 수 있으며 시스템에 **Comment**를 추가한 **Writer**와 **Date**가 저장되고 **User**가 확인할 수 있게 한다.

iii. developer



1. Developer는 시스템에 id와 password를 입력하고 system은 이것을 name과 password로 받아와 해당 계정이 있으면 로그인 후 home으로 이동  
해당 계정이 없으면 시스템이 로그인 실패를 알린다.
2. Developer는 자신에게 할당된 Issue를 확인하여 코드를 수정하고 상태를 fixed로 수정할 수 있다.
3. Developer는 Issue에 Comment를 추가할 수 있으며 시스템에 Comment를 추가한 Writer와 Date가 저장되고 User가 확인할 수 있게 한다.

iv. tester

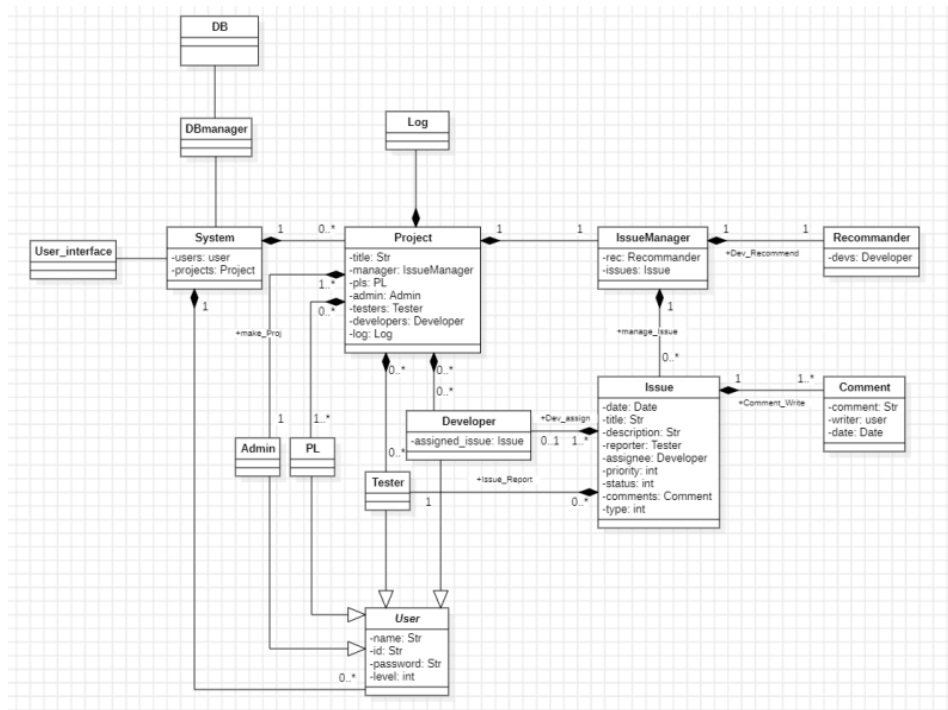


1. Tester는 시스템에 id와 password를 입력하고 system은 이것을 name과 password로 받아와  
해당 계정이 있으면 로그인 후 home으로 이동  
해당 계정이 없으면 시스템이 로그인 실패를 알린다.
2. Tester는 Issue를 추가할 수 있으며 title을 작성하고 priority와 type를 선택합니다. 선택할 수 있는 status는 new로 고정되어있습니다. Issue를 추가하면 시스템에게 Issue가 생성되었다는 응답을 받습니다.
3. Tester는 Dev가 fixed한 Issue를 확인하고 상태를 resolved로 바꿀 수 있습니다. 상태를 변경하면 시스템에게 변경이 완료되었다는 응답을 받습니다.

### 3. 설계

#### a. 클래스 다이어그램

##### i. 초기 클래스 다이어그램



##### ii. 최종 클래스 다이어그램 (화질 이슈로 별도의 사진 파일 첨부하였음.)



### iii. 설명

#### 1. Domain

도메인을 이루는 클래스들로 MVC의 **Model**부분에 해당된다. 이슈관리 시스템의 주요 데이터들을 다루며 데이터베이스 테이블과 매핑되는 클래스.

- a. **MemberClass** - 시스템을 사용하는 멤버들의 데이터를 가지고 있으며 멤버 객체를 생성하는데 필요한 생성자와 내부 데이터를 가져오는데 필요한 **getter** 메서드가 포함되어 있다.
- b. **ProjectClass** - 해당 프로그램에서 관리하는 프로젝트의 데이터를 가지고 있으며 프로젝트 객체를 생성하는데 필요한 생성자와 **getter** 메서드를 포함하고 있다.
- c. **IssueClass** - 프로젝트에 추가되는 이슈들의 데이터를 가지고 있으며 생성자, **getter** 메서드와 이슈 상태를 정의하는데 필요한 **enum**을 포함하고 있다.
- d. **CommentClass** - 이슈에 추가되는 코멘트의 데이터를 가지고 있으며 생성자와 **getter** 메서드를 포함하고 있다.

#### 2. Controller

크게 **Domain**에 해당하는 클래스들에 대한 **HTTP** 요청을 해석해서 **Model**과 **View**에게 알리는 **Class**들의 집합. MVC의 **Controller**부분에 해당된다.

- a. **MemberController** - 멤버들에 대한 **Api**를 다루며 로그인, 사용자 추가, 사용자 정보 불러오기등의 **View**의 요청을 받아 해석한다.



- b. **ProjectController** - 프로젝트에 대한 **Api**를 다루며 **View**의 **Project** 생성 및 정보에 대한 요청을 해석한다.
- c. **IssueController** - 프로젝트에 포함되는 이슈에 대한 **Api**를 다루며 이슈를 생성하거나 생성되어있는 이슈의 정보 및 트랜드를 확인하는 요청등을 해석한다.
- d. **CommentController** - 이슈에 포함되는 코멘트에 대한 **Api**를 다루며 코멘트를 추가하거나 코멘트 나열에 대한 요청을 해석한다.

### 3. Service

비즈니스 로직을 다루는 클래스로 데이터 베이스에 접근하여 **controller**에서 요청한 기능을 수행한다.

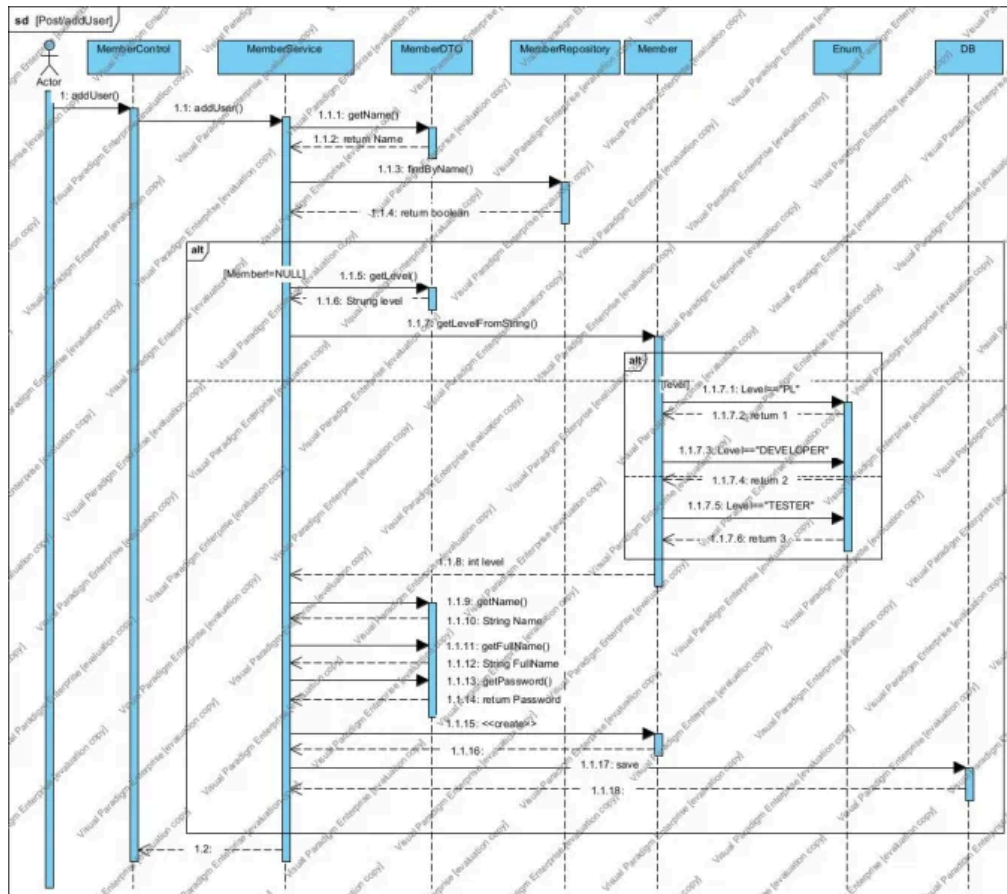
- a. **MemberService** - login, adduser, getmember등의 멤버에 관련된 기능을 구현한 메서드를 가지고 있다.
- b. **ProjectService** - addproject, getproject등의 프로젝트에 관련된 기능을 구현한 메서드를 가지고 있다.
- c. **IssueService** - addissue, getissue, setstate, suggestassignee등의 이슈에 관련된 기능을 구현한 메서드를 가지고 있다.
- d. **CommentService** - addcomment, getcomment등의 코멘트에 관련된 기능을 구현한 메서드를 가지고 있다.

### 4. Repository

데이터베이스 접근을 담당하며 **JPA** 모듈을 사용해 데이터베이스를 관리한다.

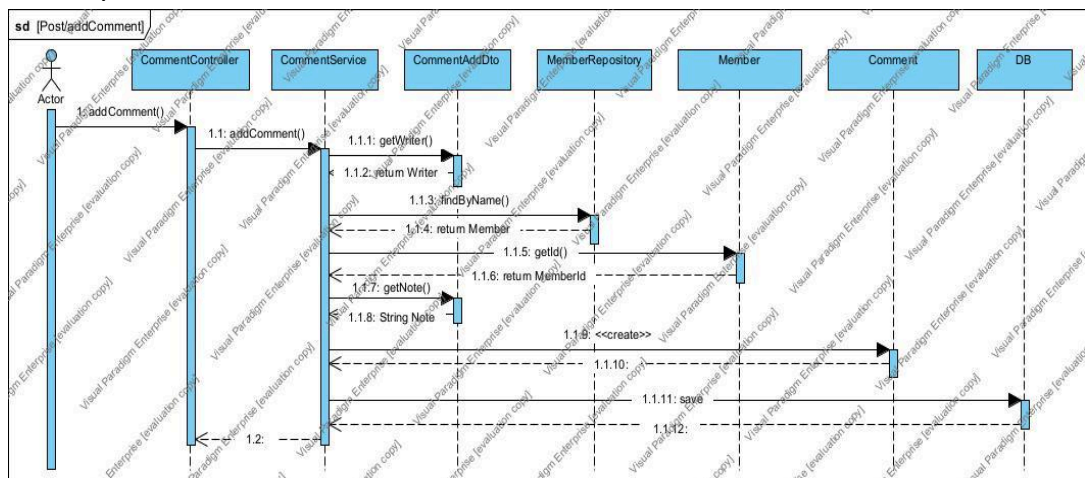
- a. **MemberRepository** - 데이터베이스의 **Member Table**에 **data**를 추가 및 수정한다.
- b. **ProjectRepository** - 데이터베이스의 **Project Table**에 **data**를 추가 및 수정한다.
- c. **IssueRepository** - 데이터베이스의 **Issue Table**에 **data**를 추가 및 수정한다.
- d. **CommentRepository** - 데이터베이스의 **Comment Table**에 **data**를 추가 및 수정한다.

- b. 시퀀스 다이어그램
  - post/addUser



Actor가 MemberController에 정의된 addUser api를 호출하여 addUser함수를 실행한다. 곧바로 MemberDTO에 클라이언트에서 보낸 추가될 유저 정보가 주입되며, MemberController는 MemberDTO를 매개변수로 MemberService의 addUser를 호출한다. MemberService에서는 받은 MemberDTO의 정보에서 User의 Name이 이미 MemberRepository에 존재하는지 중복을 확인한 후, 확인 여부를 boolean으로 리턴한다. 중복이 되지 않는다면 MemberDTO의 정보를 기반으로 Member객체를 새로 create한 뒤에 DB에 저장한다.

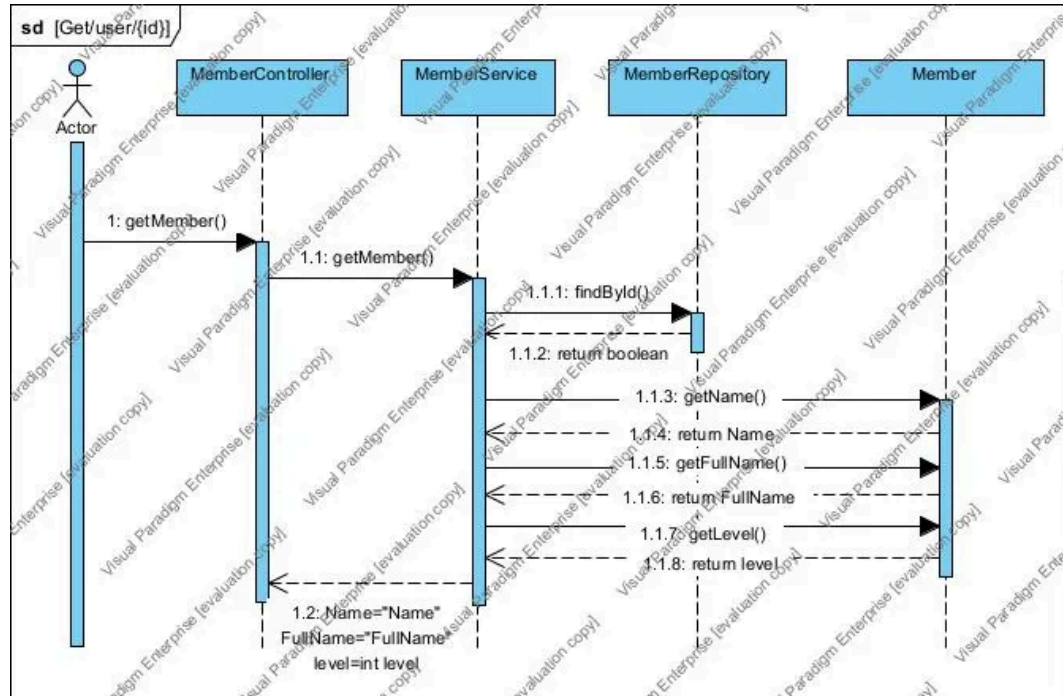
#### - post/addComment



Actor가 CommentController에 정의된 addComment api를 호출하여 addComment함수를 실행한다. 곧바로 CommentAddDto에 클라이언트에서 보낸 추가될 코멘트 정보가 주입되며, CommentController는 CommentAddDTO를

매개변수로 **CommentService**의 **addComment**함수를 호출한다.  
**CommentService**에서는 받은 **Dto**정보를 기반으로 **Comment** 객체를 새로 **create**한 뒤에 **DB**에 저장한다.

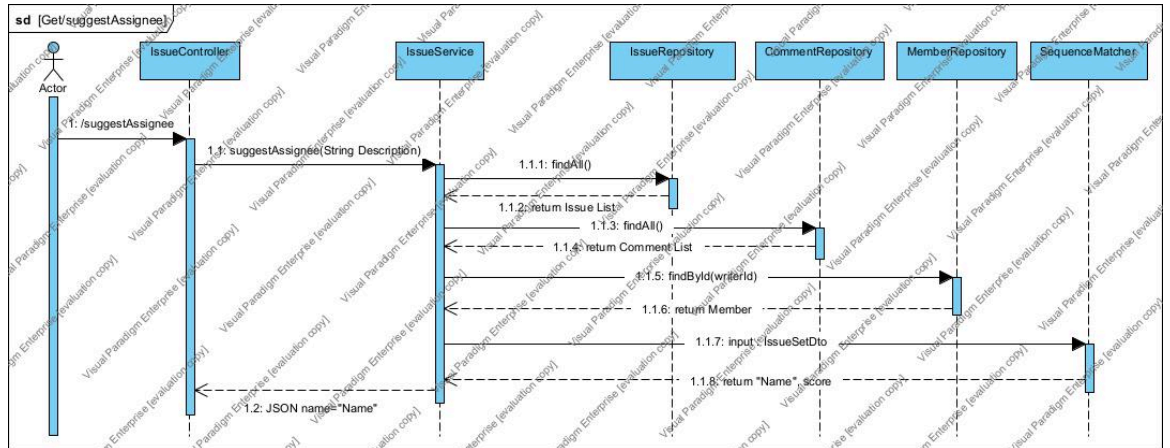
- **Get/user/{id}**



위 시퀀스 다이어그램은 **Get/user/{id}**의 사용자가 특정 ID를 가진 멤버의 정보를 조회하는 과정을 표현하며, 그 흐름은 다음과 같다.

- 1) **Actor**: 사용자가 **getMember()** 메소드를 호출한다.
- 2) **MemberController**: **getMember()** 메소드를 호출하여 **MemberService**로 요청을 전달한다.
- 3) **MemberService**: **findById()**를 호출하여 **MemberRepository**에서 해당 ID를 가진 멤버를 검색하고 결과를 반환받는다. 다음 **getName()**을 호출하여 멤버의 이름을 가져오고 이름을 반환받는다. 다음 **getFullName()**을 호출하여 멤버의 전체 이름을 가져오고 전체 이름을 반환받는다. 그리고 **getLevel()**을 호출하여 멤버의 레벨을 가져오고 레벨을 반환받는다.
- 4) **MemberRepository**: **findById(id)**를 호출하여 특정 ID를 가진 멤버를 검색하고 **boolean**으로 결과를 반환한다.
- 5) **Member**: **getName()**, **getFullName()**, **getLevel()** 메서드를 호출하여 각각 멤버의 이름/전체 이름/레벨을 가져오고 반환한다.
- 6) **MemberService**: **Name=Name, FullName=FullName, level=int level** 결과를 **MemberController**로 반환한다.

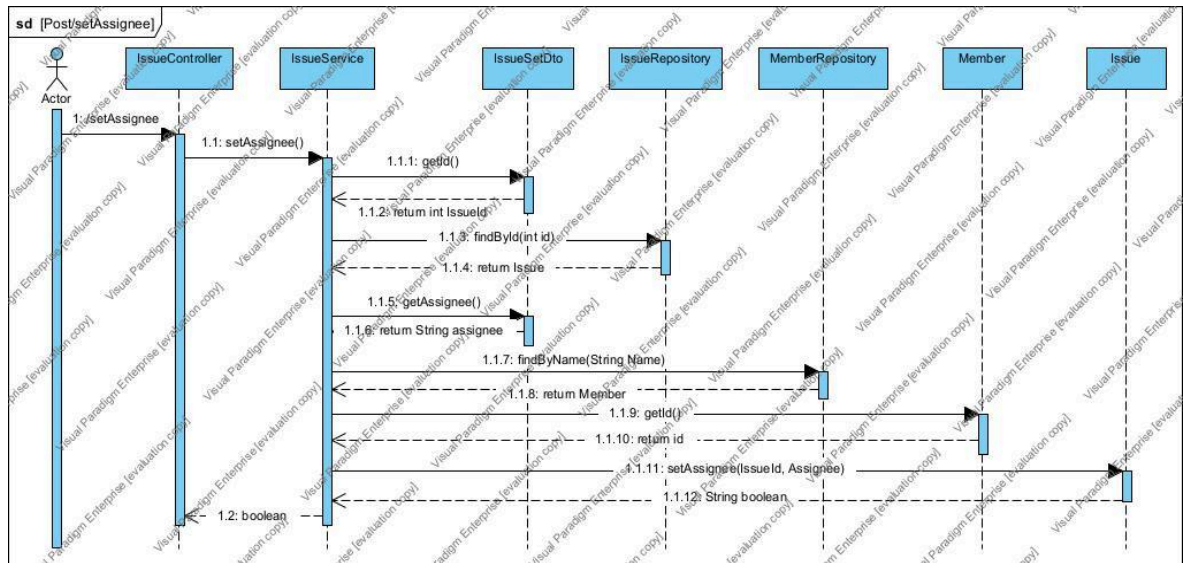
- **Get/suggestAssignee**



위 시퀀스 다이어그램은 Get/suggestAssignee의 사용자가 특정 이슈에 대해 담당자를 추천받는 과정을 표현하며, 그 흐름은 다음과 같다.

- 1) Actor: 사용자가 /suggestAssignee 엔드포인트를 호출한다.
- 2) IssueController: suggestAssignee(String Description) 메소드를 호출하여 IssueService로 요청을 전달한다.
- 3) IssueService: findAll()을 호출하여 IssueRepository에서 모든 이슈를 검색하고 이슈 리스트를 반환받는다. 다음 findAll()을 호출하여 CommentRepository에서 모든 댓글을 검색하고 댓글 리스트를 반환받는다. 다음 findByld(writerId)를 호출하여 MemberRepository에서 댓글 작성자를 검색하고 멤버 객체를 반환받는다. 다음 input을 통해 IssueSetDto 객체를 SequenceMatcher로 전달하고 추천된 담당자의 이름을 반환받는다.
- 4) IssueRepository: findAll()을 호출하여 모든 이슈를 검색하고 이슈 리스트를 반환한다.
- 5) CommentRepository: findAll()을 호출하여 모든 댓글을 검색하고 이슈 리스트를 반환한다.
- 6) MemberRepository: findByld(writerId)을 호출하여 특정 작성자를 검색하고 멤버 객체를 반환한다.
- 7) SequencMatcher: input을 통해 IssueSetDto 객체를 입력받고 추천된 담당자 이름과 점수를 반환한다.
- 8) IssueService: JSON name = Name의 결과를 IssueController로 반환한다.

- Post/setAssignee

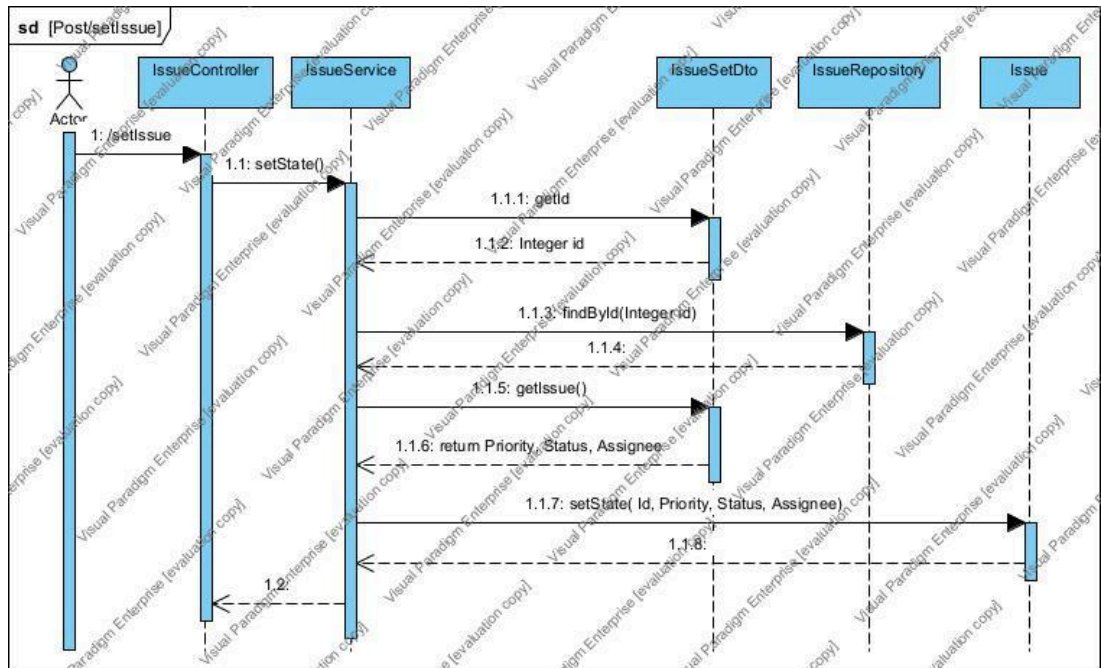


위 시퀀스 다이어그램은 **Post/setAssignee**의 특정 이슈에 대해 담당자를 지정하는 과정을 표현하며, 그 흐름은 다음과 같다.

- 1) **Actor**: 사용자가 'setAssignee' 메소드를 호출한다.
- 2) **IssueController**: setAssignee() 메소드를 호출하여 **IssueService**로 요청을 전달한다.
- 3) **IssueService**: getId()를 호출하여 **IssueSetDto** 객체를 가져오고, **IssueID**를 반환받는다. 다음 **findById(int id)**를 호출하여 **IssueRepository**에서 이슈를 검색하고, 이슈 객체를 반환받는다. 그리고 **getAssignee**를 호출해 현재 지정된 담당자를 가져와 정보를 반환받고 **findByName(String name)**을 호출하여 **MemberRepository**에서 해당 멤버를 검색한 뒤 멤버의 객체를 반환받는다.
- 4) **IssueRepository**: getId()를 호출하여 이슈 ID를 가져와 반환한다.
- 5) **MemberRepository**: findByName(String Name)을 호출하여 멤버를 검색하고 반환한다.
- 6) **Member/Issue**: setAssignee(IssueId, Assignee)를 호출하여 새로운 담당자를 지정하고 **return boolean**으로 지정 성공 여부를 반환한다.
- 7) **IssueService**: 최종적으로 그 결과를 **IssueController**로 반환한다.

- **Post/setIssue**



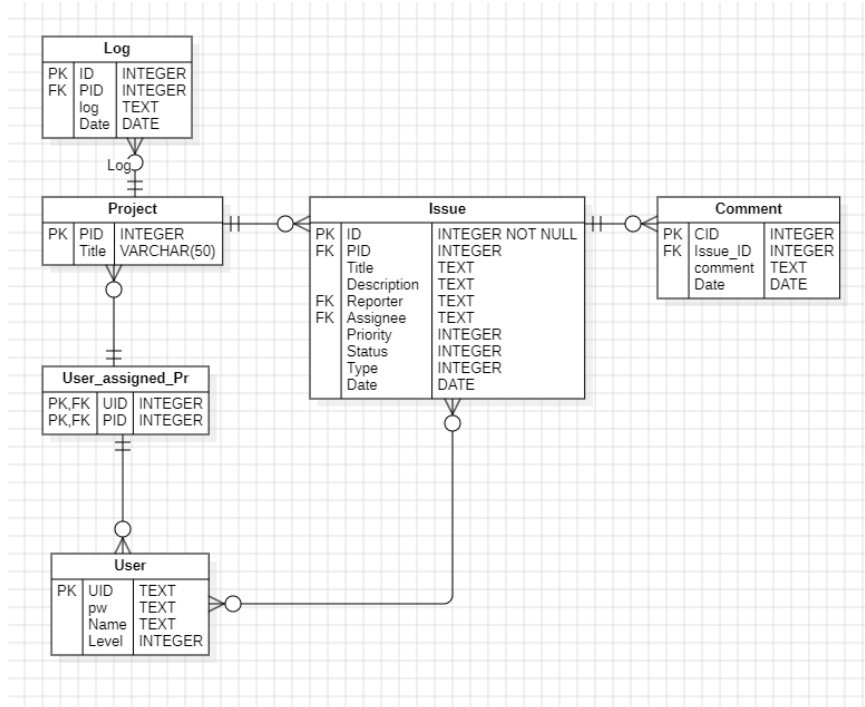


위 시퀀스 다이어그램은 **Post/setIssue**의 특정 이슈의 우선순위와 상태, 담당자를 변경하는 과정을 표현하며, 그 흐름은 다음과 같다.

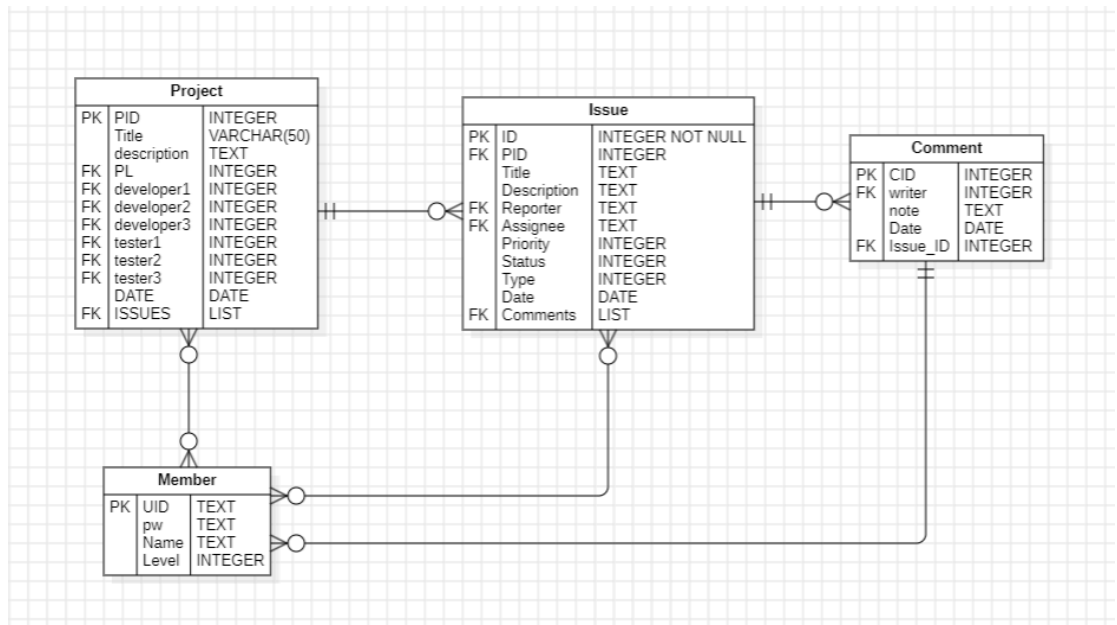
- 1) 사용자가 **setIssue()**를 **IssueController**에 전송한다.
- 2) **IssueController**는 요청을 수신하고 **IssueService**의 **setState()** 메서드를 호출한다.
- 3) **IssueService**는 **IssueSetDto** 객체의 **getId()** 메서드를 호출하여 새 이슈의 ID를 가져온다.
- 4) **IssueService**는 **IssueRepository**의 **findById(Integer id)** 메서드를 호출하여 지정된 ID를 가진 기존 이슈 객체를 가져온다.
- 5) **IssueService**는 기존 이슈 객체의 **getissue()** 메서드를 호출하여 이슈 데이터를 가져온다.
- 6) **IssueService**는 **IssueRepository**의 **setState(Id, Priority, Status, Assignee)** 메서드를 호출하여 이슈 상태를 업데이트한다.
- 7) **IssueService**는 **IssueController**에 성공 응답을 반환한다.
- 8) **IssueController**는 사용자에게 성공 응답을 반환한다.

c. ER 다이어그램

i. 초기 ER 다이어그램



## ii. 수정된 ER 다이어그램



## d. GRASP 패턴 원칙의 적용

아래는 GRASP 패턴이 어떻게 프로젝트에 적용되었는지 서술한다.

- Information expert** 개체는 프로젝트 내에서 **Domain** 패키지와 파일들로 구성되어 있으며, 역할을 받은 **DTO**는 데이터를 다른 계층으로 전달하기 위한 객체를 포함하고, **Repository**는 도메인 객체를 저장하거나 검색하며, **Service**는 비즈니스 로직을 구현하고, **Controller**는 사용자 인터페이스와 상호작용한다.
- Creator** 개체는 프로젝트 내에서 **domain** 패키지와 각 객체의 생성자들로 구성되어 있으며, 데이터와 관련된 책임을 객체가 처리하여 응집도를 높이고 결합도를 낮추기 위해 사용되었다.

- iii. **Controller** 개체는 프로젝트 내에서 **controller** 패키지와 각 도메인에 대응되는 파일들로 구성되어 있으며, 사용자 인터페이스인 **web UI / Swing UI**와 내부 구조를 분리하고 객체에 대한 직접적인 접근을 줄이기 위해 사용되었다.
- iv. **Indirection** 개체는 프로젝트 내에서 **service**와 **repository** 패키지의 각 도메인에 대응되는 파일들로 구성되어 있으며, 이들은 **DB**와 상호작용하여 도메인 객체를 저장하거나 검색하고 상호작용하여 복잡한 작업을 수행한다.
- v. **Low coupling**은 각 객체가 밀접하게 연관된 역할들만 가지도록 기능을 함으로써 구현되며, 이는 프로젝트 내에서 **Member/Project/Issue/Comment**의 객체가 각자 사용되면서 자신의 작업이 다른 클래스에게 영향을 미치지 않는다.
- vi. **High cohesion**은 각 모듈이나 클래스가 특정 기능을 명확히 수행하고 관련 있는 책임들이 한 곳에 집중됨으로써 구현되며, 이는 프로젝트 내에서 **Domain/DTO/Repository/Service/Controller**가 명확히 분리되어 실행됨으로써 구현되었다.
- vii. **Polymorphism**은 다양한 형태의 객체들이 동일한 인터페이스로 동일하게 작동함으로써 구현되며, 이는 **Controller** 내의 동일한 **API**가 다양한 형식의 입력을 받고도 동일하게 작동함으로써 구현되었다.
- viii. **Pure fabrication** 개체는 도메인 객체에 속하지 않는 책임을 구현하기 위해 존재하며, 이는 **Service** 패키지와 휘하 클래스들이 시스템의 특정 역할을 수행하여 사용자가 이용할 수 있는 서비스를 구동함으로써 구현되었다.
- ix. **Protected variations** 개체는 변화로부터 시스템의 다른 부분을 보호하고 영향을 최소화하기 위해 존재하며, 이는 시스템이 **Controller**의 **API**를 통해 프론트엔드/**UI**와 통합되어 작동함으로써 구현되었다.

#### 4. 주요 기능 구현 (두 UI는 하나의 **Model** 위에서 동일한 로직으로 구동됨)

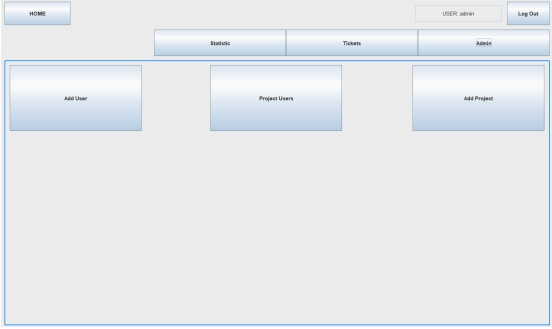
##### a. 로그인

<div style="text-align: center; font-weight: bold; margin-bottom: 10px;">Issue Ticketing System</div> <div style="border: 1px solid black; padding: 10px; margin-bottom: 10px;"> <div style="border-bottom: 1px solid black; padding-bottom: 5px; margin-bottom: 5px;">ID</div> <div style="border-bottom: 1px solid black; padding-bottom: 5px;">Password</div> </div> <div style="text-align: right; margin-top: 10px;"> <input type="button" value="Login"/> </div>	<div style="border: 1px solid gray; padding: 10px; margin-bottom: 10px; background-color: #f0f0f0;"> <div style="display: flex; justify-content: space-between; align-items: center; border-bottom: 1px solid gray; padding-bottom: 5px;"> <span>Login</span> <span>— □ ×</span> </div> <div style="text-align: center; margin-top: 10px;">Issue Ticketing System</div> <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 10px;"> <input style="width: 150px;" type="text"/> <div style="border: 1px solid gray; padding: 5px 10px; background-color: #add8e6;">Sign in</div> </div> </div>
Web UI (html css js)	Swing UI

서비스에 처음 접속 시 로그인 화면이 발생하며, 사용자 ID와 비밀번호를 작성하고 버튼을 누르면 해당 계정으로 서비스를 이용할 수 있게 된다.


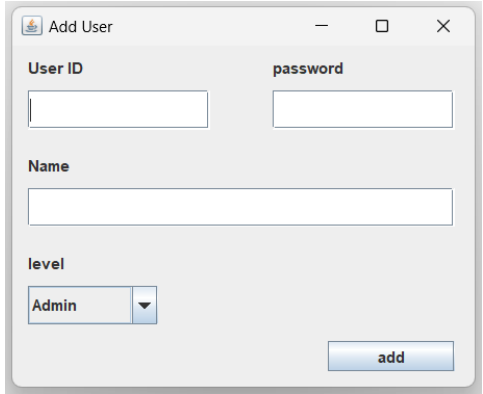
##### b. admin 기능



<div data-bbox="352 215 440 253">Home</div> <h3 data-bbox="531 277 671 309">Admin Page</h3> <div data-bbox="491 338 715 367">Add User</div> <div data-bbox="491 398 715 427">Add Project</div> <div data-bbox="491 459 715 488">User Management</div>	
Web UI (html css js)	Swing UI

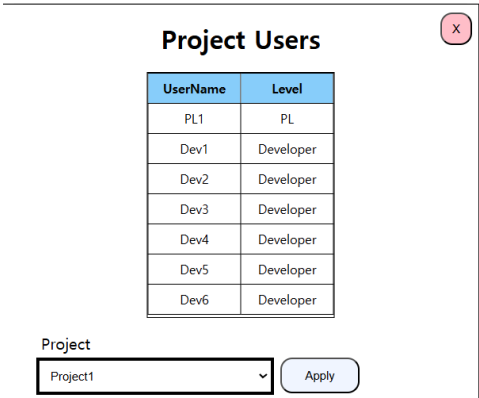
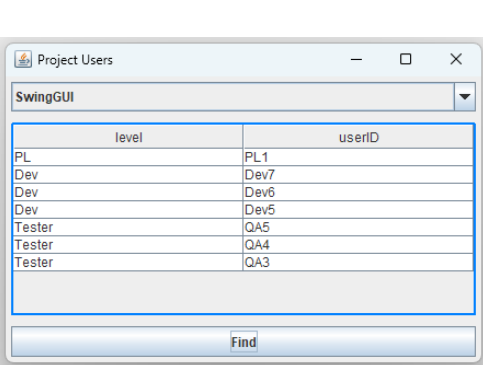
사용자가 **Admin** 계정으로 서비스를 접속하고 상단의 **Admin** 버튼을 누를 경우 관리자 기능인 계정 추가/프로젝트 추가/사용자 관리를 할 수 있게 된다. 이 버튼은 **Admin** 계정으로 접속 시에만 작동하며, 그 외에는 작동하지 않는다.

#### c. 계정 추가

	
Web UI (html css js)	Swing UI

계정 추가는 **Admin** 권한을 가진 사용자가 추가하며, 사용자의 이름, ID, 비밀번호, 권한을 입력함으로써 추가된다.

#### d. 프로젝트 별 사용자 목록 조회

	
Web UI (html css js)	Swing UI

특정 프로젝트를 선택하고 해당 프로젝트에 할당된 사용자들의 목록을 확인할 수 있다.

e. 프로젝트 추가

<div><h3>Project Add</h3><div>제목 제목을 입력하세요.</div><div>내용</div><div><div>PL PL1</div><div>Dev Dev1 Dev2 Dev3</div><div>Tester QA1 QA2 QA3</div></div><div>Add</div></div>	<div><h3>Add new project</h3><div>Title aaa</div><div>Description aaa</div><div><div>PL1</div><div>Dev1 Dev2 Dev3</div><div>QA1 QA2 QA3</div></div><div>add</div></div>
Web UI (html css js)	Swing UI

프로젝트 추가는 Admin 권한을 가진 사용자가 추가하며, 프로젝트의 제목, 내용, 사용자들을 입력함으로써 추가된다. 이때 개발자와 테스터는 항상 3명을 입력해야 한다.

f. 프로젝트 확인

<div><h3>Project List (Home)</h3><div>Statistics Admin</div><table><thead><tr><th>ProjectTitle</th><th>Description</th><th>Date</th></tr></thead><tbody><tr><td>Project1</td><td>This is Project1</td><td>2024/08/01</td></tr><tr><td>Project2</td><td>This is Project 2.</td><td>2024/06/01</td></tr></tbody></table></div>	ProjectTitle	Description	Date	Project1	This is Project1	2024/08/01	Project2	This is Project 2.	2024/06/01	<div><h3>HOME</h3><div>Statistics Tickets Admin</div><div>Log Out</div><div>Project1 This is Project1 2024/08/01</div><div>Project2 This is Project 2. 2024/06/01</div></div>
ProjectTitle	Description	Date								
Project1	This is Project1	2024/08/01								
Project2	This is Project 2.	2024/06/01								
Web UI (html css js)	Swing UI									


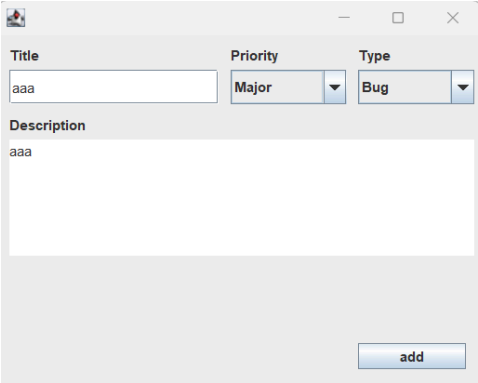
사용자는 home에서 프로젝트의 목록을 확인할 수 있다. 이때 ProjectDto를 선택하면(프론트엔드)/프로젝트를 선택하고 Tickets를 누르면(Swing UI) 해당 프로젝트 내의 이슈들을 확인할 수 있다.

g. 이슈 브라우즈 및 검색

<div><h3>Issue Information</h3><div>Home Statistics Admin</div><table><thead><tr><th>IssueId</th><th>Title</th><th>Priority</th><th>Status</th><th>Type</th><th>Reporter</th><th>Assignee</th><th>Date</th></tr></thead><tbody><tr><td>3</td><td>Issue 1</td><td>Minor</td><td>Assigned</td><td>Bug</td><td>admin</td><td>Dev1</td><td>2024/08/01</td></tr><tr><td>4</td><td>Issue 2</td><td>Blocker</td><td>Fixed</td><td>Bug</td><td>admin</td><td>Dev2</td><td>2024/08/01</td></tr><tr><td>5</td><td>Issue 3</td><td>Major</td><td>New</td><td>Bug</td><td>admin</td><td>N/A</td><td>2024/08/01</td></tr><tr><td>6</td><td>Issue 4</td><td>Trivial</td><td>Assigned</td><td>Task</td><td>admin</td><td>Dev3</td><td>2024/08/01</td></tr></tbody></table><div>View search Search Reset New Add</div></div>	IssueId	Title	Priority	Status	Type	Reporter	Assignee	Date	3	Issue 1	Minor	Assigned	Bug	admin	Dev1	2024/08/01	4	Issue 2	Blocker	Fixed	Bug	admin	Dev2	2024/08/01	5	Issue 3	Major	New	Bug	admin	N/A	2024/08/01	6	Issue 4	Trivial	Assigned	Task	admin	Dev3	2024/08/01	<div><h3>HOME</h3><div>Statistics Tickets Admin</div><div>Log Out</div><div>IssueId Title Priority Status Type Reporter Assignee Date</div><div>3 Issue 1 Minor Assigned Bug admin Dev1 2024/08/01</div><div>4 Issue 2 Blocker Fixed Bug admin Dev2 2024/08/01</div><div>5 Issue 3 Major New Bug admin N/A 2024/08/01</div><div>6 Issue 4 Trivial Assigned Task admin Dev3 2024/08/01</div></div>
IssueId	Title	Priority	Status	Type	Reporter	Assignee	Date																																		
3	Issue 1	Minor	Assigned	Bug	admin	Dev1	2024/08/01																																		
4	Issue 2	Blocker	Fixed	Bug	admin	Dev2	2024/08/01																																		
5	Issue 3	Major	New	Bug	admin	N/A	2024/08/01																																		
6	Issue 4	Trivial	Assigned	Task	admin	Dev3	2024/08/01																																		
Web UI (html css js)	Swing UI																																								

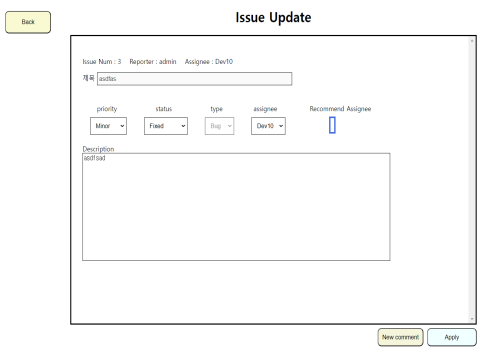
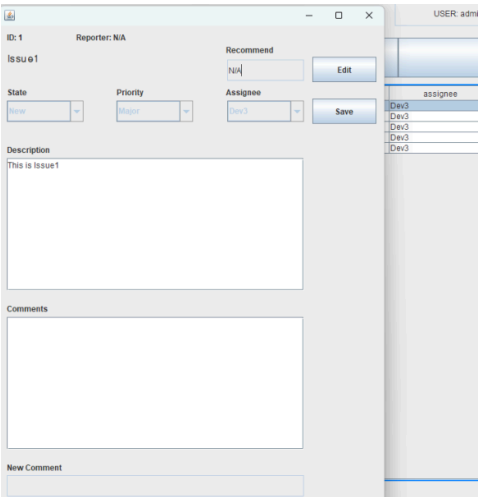
프로젝트를 선택하면 해당 프로젝트 내의 이슈들을 확인할 수 있다. 이때 하단에서 이슈의 **Title/Priority/Status/Type/Reporter/Assignee** 검색 유형을 선택하고 문자열로 입력하면 일치하는 이슈들이 검색되며, 일치하는 이슈가 없으면 빈 이슈칸을 보여준다. **Swing UI**의 경우 상단의 바를 선택하여 해당 기준으로 이슈를 정렬할 수 있다.

#### h. 이슈 등록

 <p><b>Issue Add</b></p> <p>Reporter: admin</p> <p>제목 Title</p> <p>priority status type</p> <p>Major New Bug</p> <p>Description</p> <p>Add</p>	 <p><b>Title</b> <b>Priority</b> <b>Type</b></p> <p>aaa Major Bug</p> <p><b>Description</b></p> <p>aaa</p> <p>add</p>
Web UI (html css js)	Swing UI


하단의 **Add Issue** 버튼을 선택하면 이슈를 등록할 수 있으며, 이슈의 제목, **Priority/Type**/설명을 입력함으로써 추가된다. 이슈의 **Status**는 기본적으로 **New**로 배정되고 이슈 등록 후 후술할 상세 정보에서 변경이 가능하다.

#### i. 이슈 상세 정보 확인 및 상태 변경

 <p><b>Issue Update</b></p> <p>Issue Num: 3 Reporter: admin Assignee: Dev10</p> <p>제목 adf123</p> <p>priority status type assignee Recommend Assignee</p> <p>Minor Fixed Bug Dev10</p> <p>Description adf123</p> <p>New comment Apply</p>	 <p>ID: 1 Reporter: N/A</p> <p>Issue #1</p> <p>State Priority Assignee</p> <p>New Major Dev1</p> <p>Description This is issue1</p> <p>Comments</p> <p>New Comment</p> <p>USER: admin</p>
Web UI (html css js)	Swing UI

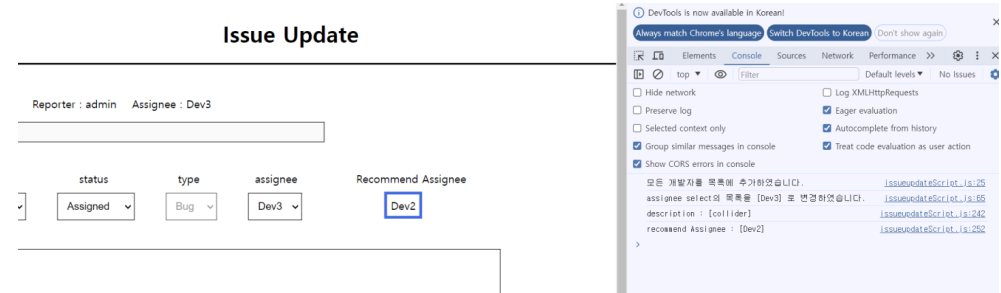
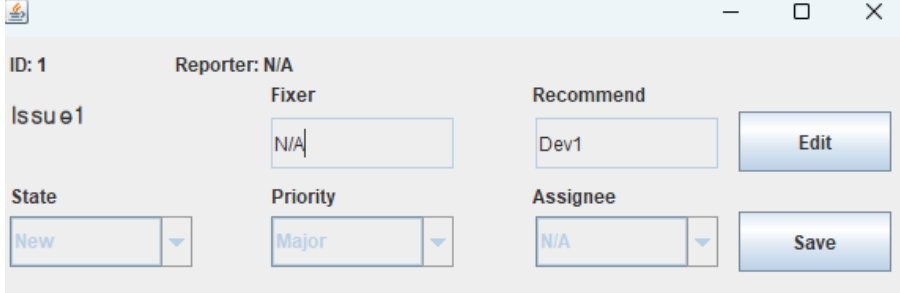
생성된 이슈를 선택하면 이슈의 상태 정보를 확인할 수 있으며, 이곳에서 이슈의 제목/**Priority/Status/Assignee**를 변경할 수 있다. 이때 **Assignee**를 할당한 이후에 **Status**의 변경이 가능해지고 하단의 **Apply** 버튼을 누르면 변경이 완료된다.

#### j. 이슈 코멘트 추가

	issue edit창에서 comment 추가 가능
Web UI (html css js)	Swing UI

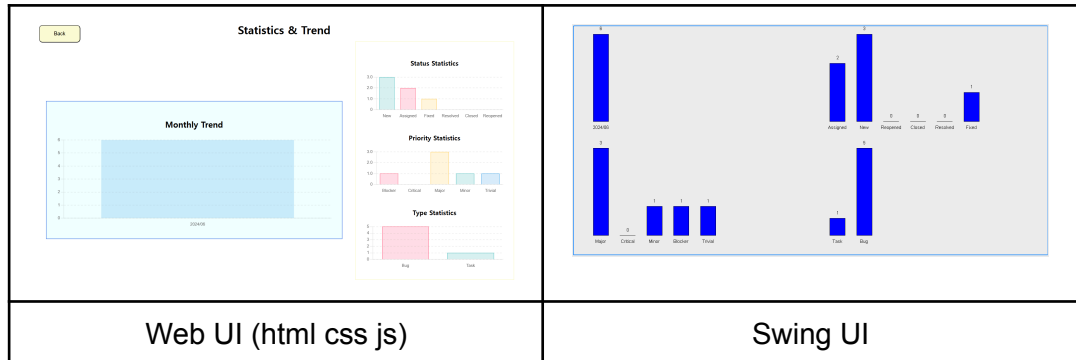
이슈 상태 정보 하단의 **New comment** 버튼을 선택하면 코멘트를 추가할 수 있으며, 코멘트의 내용을 작성하고 **Add** 버튼을 누르면 코멘트가 등록된다. 등록된 코멘트는 이슈 상태 정보의 하단에서 확인할 수 있다.

#### k. 이슈 이력을 활용한 assignee 자동 추천 기능

	Web UI (html css js)
	Swing UI

이슈 등록 후 **assignee**를 할당하기 전 추천 기능은 현 이슈와 이전 이슈들의 내용/코멘트의 유사도값을 측정하여 가장 많이 배당된 개발자를 추천해준다. 이 기능은 이슈와 코멘트에 대한 **DB**가 충분히 쌓여있어야 의도한 대로 작동한다.

#### l. 이슈 통계 분석



사용자가 상단의 **Statistics** 버튼을 누르면 지금까지 생성된 이슈들의 통계를 보여준다. 이 이슈 통계에는 월별 이슈 발생 횟수와 이슈의 **Status/Priority/Type** 통계를 막대 그래프 형태로 보여준다.

## 5. 테스트 수행 내역

### a. 테스트 케이스

#### i. Project Controller

1. 프로젝트 추가 성공: 프로젝트의 추가가 성공했는지 확인한다.
2. 프로젝트 추가 실패: 프로젝트의 제목이 중복되어 추가가 실패한 경우 이를 확인한다.
3. 프로젝트 **get** 성공: 특정 프로젝트의 **id**로 프로젝트를 가져왔는지 확인한다.
4. 프로젝트 **get** 실패: **id**로 가져오고자 하는 프로젝트가 존재하지 않아 프로젝트를 가져오지 못했을 경우 이를 확인한다.
5. 프로젝트 **get by title** 성공: 특정 프로젝트의 제목으로 프로젝트를 가져왔는지 확인한다.
6. 프로젝트 **list get** 성공: 프로젝트의 전체 리스트를 **JSON** 문자열 형태로 가져왔는지 확인한다.

#### ii. Member Controller

1. 로그인 성공: 임의의 사용자가 로그인에 성공했는지 확인한다.
2. 로그인 실패: 잘못된 **name**을 입력하여 사용자가 로그인에 실패했는지 확인한다.
3. 로그인 실패: 잘못된 비밀번호를 입력하여 사용자가 로그인에 실패했는지 확인한다.
4. 멤버 추가 성공: 사용자의 추가가 성공했는지 확인한다.
5. 멤버 추가 실패: 사용자를 추가할 때 중복된 **name**을 입력하여 추가가 실패한 경우 이를 확인한다.
6. 멤버 **get** 성공: 특정 사용자의 **name/fullName**으로 해당 사용자를 가져왔는지 확인한다.
7. 멤버 **get** 실패: 잘못된 **name/fullName**으로 해당 사용자를 가져오지 못하였는지 확인한다.

#### iii. Issue Controller

1. 이슈 추가 성공: 이슈의 추가가 성공했는지 확인한다.
2. 이슈 추가 실패: 이슈의 **title**이 중복되어 이슈 추가가 실패했는지 확인한다.
3. 이슈 **get** 성공: 특정 이슈의 **id**로 이슈를 가져왔는지 확인한다.
4. 이슈 **get** 실패: **id**로 가져오고자 하는 이슈가 존재하지 않아 이슈를 가져오지 못했을 경우 이를 확인한다.

5. 이슈 **list get** 성공: 존재하는 모든 이슈를 가져왔는지 확인한다.
6. 프로젝트 이슈 **list get** 성공: 특정 프로젝트의 **id**로 해당 프로젝트의 이슈를 가져왔는지 확인한다.
7. 이슈 상태 설정 성공: 변경한 이슈의 **priority, status, assignee**가 정상적으로 해당 이슈에 적용되었는지 확인한다.
8. 이슈 통계 **get** 성공: 전체 이슈의 통계를 가져왔는지 확인한다.
9. 이슈 트렌드 **get** 성공: 전체 이슈의 트렌드를 가져왔는지 확인한다.

iv. **Comment Controller**

1. 코멘트 추가 성공: 코멘트의 추가가 성공했는지 확인한다.
2. 코멘트 **list get** 성공: 임의의 이슈 **id**로 해당 이슈의 코멘트를 가져왔는지 확인한다.

v. **UserAssign Controller**

1. 유저 프로젝트에 할당 성공: 프로젝트에서 사용자의 할당이 성공했는지 확인한다.
2. 유저 프로젝트에 할당 실패: 프로젝트에 사용자가 이미 할당되어 있어 실패한 경우 이를 확인한다.
3. 유저 프로젝트에 할당 실패: 잘못된 **title**을 입력하여 사용자 할당에 실패한 경우 이를 확인한다.
4. 유저 프로젝트에 할당 실패: 잘못된 **name**을 입력하여 사용자 할당에 실패한 경우 이를 확인한다.
5. 할당 정보 **get** 성공: 프로젝트에 할당된 사용자들의 정보를 가져왔는지 확인한다.
6. 할당 정보 **get** 실패: 잘못된 **id**를 입력하여 프로젝트에 할당된 사용자들의 정보를 가져오는데 실패했는지 확인한다.
7. 프로젝트의 유저 할당 정보 **get** 성공: 특정 프로젝트가 할당된 사용자들의 정보들을 가져오는데 성공했는지 확인한다.
8. 유저의 프로젝트 할당 정보 **get** 성공: 특정 사용자가 할당된 프로젝트의 정보들을 가져오는데 성공했는지 확인한다.
9. 유저 프로젝트 할당 삭제 성공: 프로젝트에 할당된 사용자를 프로젝트에서의 제거에 성공했는지 확인한다.
10. 할당 정보 삭제 실패: 잘못된 **id**를 입력하여 프로젝트에 할당된 사용자를 프로젝트에서의 제거에 실패했는지 확인한다.

b. 테스트 결과

i. **project Controller**

1. 프로젝트 추가 성공  
프로젝트 추가에 성공했을 경우 **Project Service**에서 “true” 문자열을 **return**한다. 때문에 해당 테스트의 통과 여부를

```
assertEquals("true", response);
```

코드를 통해 검증했다.

```
Http response : true
```

2. 프로젝트 추가 실패 - 프로젝트 타이틀 중복  
프로젝트 추가에 실패했을 경우 **Project Service**에서 “true” 문자열을 **return**한다. 때문에 해당 테스트의 통과 여부를

```
assertEquals("false", response);
```

코드를 통해 검증했다.

```
Http response : false
```

### 3. 프로젝트 get 성공

프로젝트를 가져왔을 때 아래 코드를 통해서 title과 description을 비교해 테스트의 통과 여부를 검증한다.

```
.andExpect(jsonPath("$.title").value(projectRepository.findById(testProjectId).get().getTitle()))
.andExpect(jsonPath("$.description").value(projectRepository.findById(testProjectId).get().getDescription()))
```

```
Http response : {"date":"2024/06/01","tester3":"tester","developer2":"dev","tester2":"tester","developer3":"dev",
"tester1":"tester","description":"test2","developer1":"dev","title":"test2","PL":"pl2"}
```

### 4. 프로젝트 get 실패 - 존재하지 않는 프로젝트 id

프로젝트를 가져오는데 실패할 경우 Project Service는 빈 문자열을 반환하고 아래 코드를 통해 테스트 통과 여부를 검증한다.

```
assertEquals("", response);
```

```
Http response :
```

### 5. 프로젝트 get by title 성공

프로젝트의 title을 입력하면 해당 프로젝트의 id를 반환하는 기능으로 아래 코드를 통해 반환값이 테스트에 사용한 프로젝트 id와 일치하는지 검사한다.

```
assertEquals(Integer.toString(testProjectId), response);
```

```
Http response : 1
```

### 6. 프로젝트 list get 성공

프로젝트 리스트를 json 문자열 형태로 가져오는데 성공할 경우 isOk()를 사용해 테스트의 통과 여부를 검증한다. 가져온 프로젝트 리스트를 출력해 확인가능하다.

```
Http response : [{"date":"2024/06/01","tester3":"tester","developer2":"dev","tester2":"tester","developer3":"dev",
"tester1":"tester","description":"test2","id":1,"developer1":"dev","title":"test2","PL":"pl2"}]
```

## ii. Member Controller

### 1. 로그인 성공

로그인에 성공했을 경우 member service에서 "true" 문자열을 반환하기 때문에 아래 코드를 사용해 테스트 통과 여부를 검증한다.

```
assertEquals("true", response);
```

```
Http Response : true
```

### 2. 로그인 실패 - 잘못된 name

로그인에 실패했을 경우 member service에서 "false" 문자열을 반환하기 때문에 아래 코드를 사용해 테스트 통과 여부를 검증한다.

```
assertEquals("false", response);
```

```
Http Response : false
```

### 3. 로그인 실패 - 잘못된 비밀번호

로그인에 실패했을 경우 member service에서 "false" 문자열을 반환하기 때문에 아래 코드를 사용해 테스트 통과 여부를 검증한다.

```
assertEquals("false", response);
```

Http Response : false

4. 멤버 추가 성공

멤버 추가에 성공했을 경우 member service에서 “true” 문자열을 반환하기 때문에 아래 코드를 사용해 테스트 통과 여부를 검증한다.

```
assertEquals("true", response);
```

Http Response : true

5. 멤버 추가 실패 - 중복된 name

멤버 추가에 실패했을 경우 member service에서 “false” 문자열을 반환하기 때문에 아래 코드를 사용해 테스트 통과 여부를 검증한다.

```
assertEquals("false", response);
```

Http Response : false

6. 멤버 get 성공

가져올 멤버의 name과 fullname을 getmember로 가져온 멤버의 name과 fullname을 아래 코드를 사용해 비교하여 테스트 통과 여부를 검증한다.

```
.andExpect(jsonPath("$.name").value(memberRepository.findById(testMemberId).get().getName()))
.andExpect(jsonPath("$.fullName").value(memberRepository.findById(testMemberId).get().getFullName()))
```

Http Response : {"level":0,"name":"test","fullName":"test"}

7. 멤버 get 실패 - 잘못된 name

멤버 get을 실패할 경우 member service에서 빈 문자열을 반환하기 때문에 아래 코드를 사용하여 테스트의 통과 여부를 검증한다.

```
assertEquals("", response);
```

Http Response :

iii. Issue Controller

1. 이슈 추가 성공

Issue 추가에 성공했을 경우 Issue Service에서 “true” 문자열을 return한다. 때문에 해당 테스트의 통과 여부를

```
assertEquals("true", response);
```

코드를 통해 검증했다.

Http response : true

2. 이슈 추가 실패 - 중복된 title

중복된 title을 입력해 Issue 추가에 실패했을 경우 Issue Service에서 “false” 문자열을 return한다. 때문에 해당 테스트의 통과 여부를

```
assertEquals("false", response);
```

코드를 통해 검증했다.

Http response : false

3. 이슈 get 성공



이슈 id를 통해 이슈를 get할 경우

```
.andExpect(jsonPath("$.title").value(issueRepository.findById(testIssueId).get().getTitle()))
.andExpect(jsonPath("$.description").value(issueRepository.findById(testIssueId).get().getDescription()))
```

해당 코드를 사용해 받은 이슈의 제목과 내용을 비교해 올바른 이슈를 가져왔는지 확인한다. 가져온 이슈를 출력해 확인 가능하다.

```
Http response : {"date":"2024/06/01","description":"test2","reporter":"tester","assignee":"tester","title":"test2","priority":0,"type":0,"status":0}
```

#### 4. 이슈 get 실패 - 잘못된 이슈 id

잘못된 이슈 id로 이슈를 가져오는데 실패한 경우 Service에서 빈 문자열을 return하기 때문에

```
assertEquals("", response);
```

해당 코드를 사용해 테스트 통과를 검증한다.

```
Http response :
```

#### 5. 이슈 list get 성공

모든 이슈를 가져오는데 성공할 경우 isOK()를 통해 테스트 통과를 검증했고 가져온 이슈 리스트를 출력해 확인 가능하다.

```
Http response : [{"date":"2024/06/01","description":"test2","id":1,"title":"test2"}]
```

#### 6. 프로젝트 이슈 list get 성공

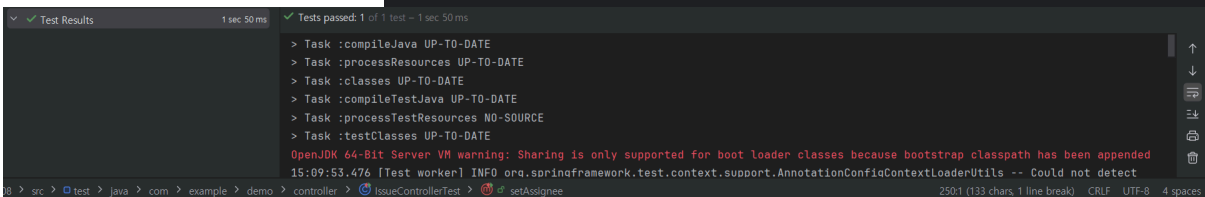
프로젝트의 모든 이슈를 가져오는데 성공할 경우 isOK()를 통해 테스트 통과를 검증했고 가져온 이슈 리스트를 출력해 확인 가능하다.

```
Http response : [{"date":"2024/06/01","issuenum":1,"description":"test2","reporter":"tester","assignee":"N/A","title":"test2","priority":0,"type":0,"status":0}]
```

#### 7. 이슈 assignee 설정 성공

assignee로 할당한 testmember의 id와 testIssue에 할당된 assignee의 id를 비교하여 테스트의 통과 여부를 검증했다.

```
assertEquals(memberRepository.findByName(testMemberName).getId(), issueRepository.findById(testIssueId).orElseThrow().getAssignee());
```



#### 8. 이슈 상태 설정 성공

issueSetDto에 입력한 변경값과 setState 실행 이후의 값을 비교하여 테스트의 통과여부를 검증했다.

```
assertEquals(issueSetDto.getPriority(), issueRepository.findById(testIssueId).get().getPriority());
assertEquals(issueSetDto.getStatus(), issueRepository.findById(testIssueId).get().getStatus());
```

```
Test Results
1 sec 80 ms
Tests passed: 1 of 1 test - 1 sec 80 ms

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava
> Task :processTestResources NO-SOURCE
> Task :testClasses

OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
15:13:45.221 [Test worker] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils -- Could not detect

src > test > java > com > example > demo > controller > IssueControllerTest > setState
```

#### 9. 이슈 통계 get 성공

이슈 통계를 가져오는데 성공할 경우 `isOK()`를 통해 테스트 통과를 검증했고 가져온 이슈 통계를 출력해 확인가능하다.

```
Http response : {"Status":{"Assigned":0,"New":1,"Reopened":0,"Closed":0,"Resolved":0,"Fixed":0},"Type":{"Task":0,"Bug":1},"Priority":{"Major":1,"Critical":0,"Minor":0,"Blocker":0,"Trivial":0}}
```

#### 10. 이슈 트렌드 get 성공

이슈 트렌드를 가져오는데 성공할 경우 `isOK()`를 통해 테스트 통과를 검증했고 가져온 이슈 트렌드를 출력해 확인가능하다.

```
Http response : {"2024/06":1}
```

### iv. Comment Controller

#### 1. 코멘트 추가 성공

Comment 추가에 성공했을 경우 Comment Service에서 “true” 문자열을 return한다. 때문에 해당 테스트의 통과 여부를 `assertEquals("true", response);` 코드를 통해 검증했고 이후 `getComment`를 사용해 추가된 Comment를 확인할 수 있다.

```
Http response : true
Http response : {"date":"2024/06/01","note":"test2","writer":"tester"}
```

#### 2. 코멘트 get 성공

Comment get에 성공했을 경우 `isOk()`를 사용해 http response를 검사했고, Service에서 return한 값을 출력해 확인가능하다.

```
Http response : [{"date":"2024/06/01","note":"test","writer":"tester"}, {"date":"2024/06/01","note":"test","writer":"tester"}, {"date":"2024/06/01","note":"test","writer":"tester"}]
```

### v. UserAssign Controller

#### 1. 유저 프로젝트에 할당 성공

유저 프로젝트 할당에 성공했을 경우 user assign Service에서 “true” 문자열을 return한다. 때문에 해당 테스트의 통과 여부를 아래 코드를 통해 검증할 수 있다.

```
assertEquals("true", response);
```

```
Http response : true
```

#### 2. 유저 프로젝트에 할당 실패 - 중복된 조합(pid,uid)

유저 프로젝트 할당에 실패했을 경우 user assign Service에서 “false” 문자열을 return한다. 때문에 해당 테스트의 통과 여부를 아래 코드를 통해 검증할 수 있다.

```
assertEquals("false", response);
```

```
Http response : false
```

#### 3. 유저 프로젝트에 할당 실패 - 잘못된 title

유저 프로젝트 할당에 실패했을 경우 **user assign Service**에서 “false” 문자열을 **return**한다. 때문에 해당 테스트의 통과 여부를 아래 코드를 통해 검증할 수 있다.

```
assertEquals("false", response);
```

```
Http response : false
```

4. 유저 프로젝트에 할당 실패 - 잘못된 name

유저 프로젝트 할당에 실패했을 경우 **user assign Service**에서 “false” 문자열을 **return**한다. 때문에 해당 테스트의 통과 여부를 아래 코드를 통해 검증할 수 있다.

```
assertEquals("false", response);
```

```
Http response : false
```

5. 할당 정보 get 성공

할당 정보를 가져오기 위해 사용한 프로젝트의 **title**과 유저 **name**을 가져온 **Json** 문자열의 **project**와 **user** 자리에 들어간 문자열과 비교하여 테스트의 통과여부를 검증한다.

```
.andExpect(jsonPath("$.project").value("test2"))  
.andExpect(jsonPath("$.user").value("pl"))
```

```
Http response : {"level":1,"project":"test2","user":"pl"}
```

6. 할당 정보 get 실패 - 잘못된 id

할당 정보 **get**에 실패할 경우 **user assign Service**에서 빈 문자열을 **return**한다. 해당 테스트의 통과 여부를 아래 코드를 통해 검증한다.

```
assertEquals("", response);
```

```
Http response :
```

7. 프로젝트의 유저 할당 정보 get 성공

프로젝트에 할당된 유저 정보를 가져오는데 성공할 경우 **isOk()**를 사용해 테스트의 통과 여부를 판단한다. **response**를 출력해 확인할 수 있다.

```
Http response : [{"level":3,"project":"test","user":"testUser"},{"level":1,"project":"test","user":"pl"},{"level":2,"project":"test","user":"dev1"}]
```

8. 유저의 프로젝트 할당 정보 get 성공

유저가 할당되어 있는 프로젝트정보를 가져오는데 성공할 경우 **isOk()**를 사용해 테스트의 통과 여부를 판단한다. **response**를 출력해 확인할 수 있다.

```
Http response : [{"level":3,"project":"test","user":"testUser"},{"level":3,"project":"test2","user":"testUser"},{"level":3,"project":"test3","user":"testUser"}]
```

9. 유저 프로젝트 할당 삭제 성공

할당 정보의 삭제에 성공했을 경우 **user assign Service**에서 “true” 문자열을 **return**한다. 때문에 해당 테스트의 통과 여부를 아래 코드를 통해 검증할 수 있다.

```
assertEquals("true", response);
```

```
Http response : true
```

10. 할당 정보 삭제 실패 - 잘못된 id

할당 정보의 삭제에 실패했을 경우 **user assign Service**에서 “false” 문자열을 **return**한다. 때문에 해당 테스트의 통과 여부를 아래 코드를 통해 검증할 수 있다.

```
assertEquals("false", response);
```

```
Http response : false
```

## 6. GitHub 프로젝트 활용

### a. GitHub 프로젝트 주소

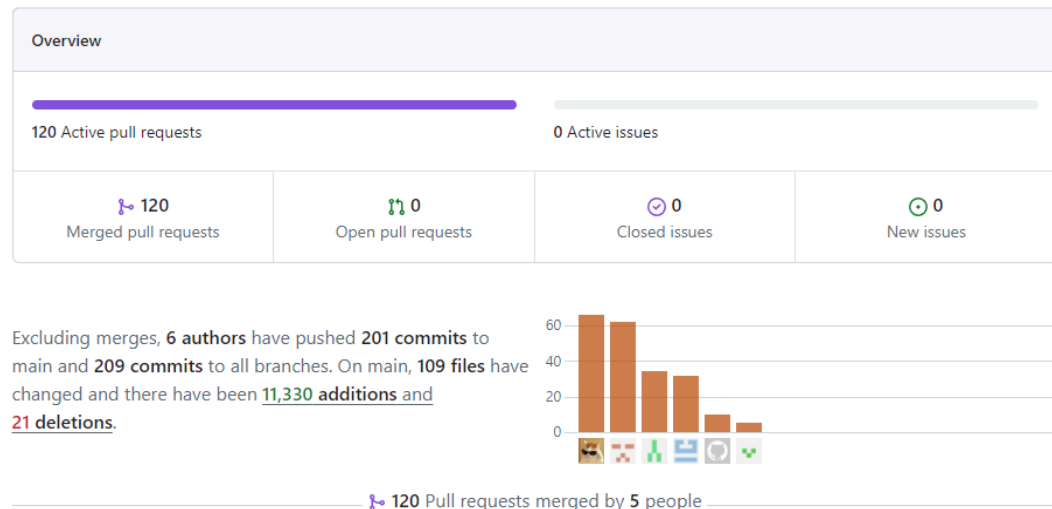
GitHub 프로젝트 주소는 아래와 같다.

<https://github.com/SE01-TeamProject/TTS08>

### b. 프로젝트 Progress History

May 2, 2024 – June 2, 2024

Period: 1 month ▾



Apr 28, 2024 – Jun 2, 2024

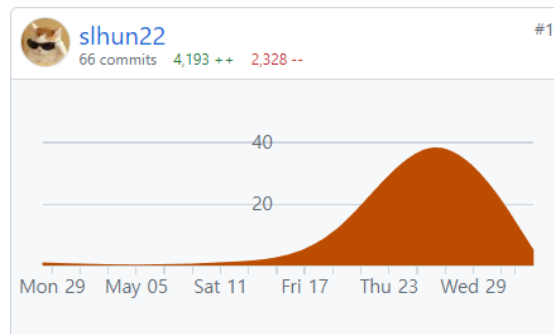
Contributions: Commits ▾

Contributions to main, excluding merge commits

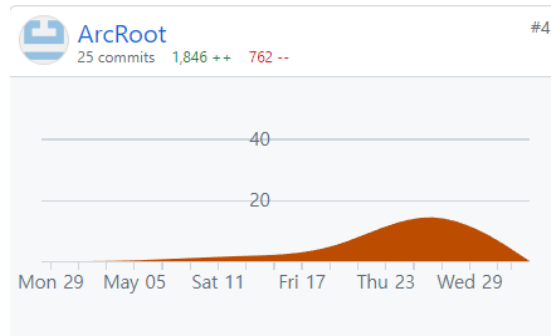


### c. 팀원별 기여

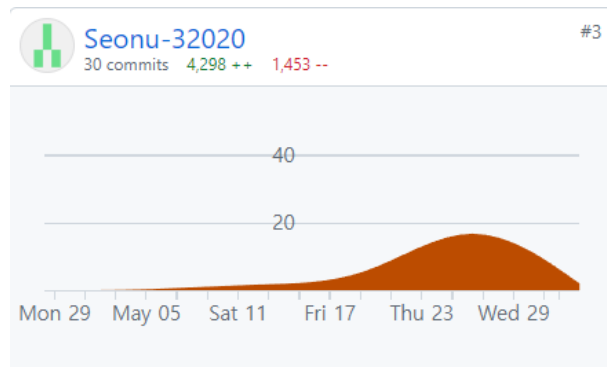
#### i. 양시훈: Web UI (html css js), Usecase리팩토링



ii. 박호근: DB, JUnit 테스트



iii. 방선우: UI, Swing



iv. 윤수용: 백엔드 메인 및 코드 작성



v. 임동재: 백엔드 보조 및 다이어그램 작성

