

Ver 1.0

# 시스템 최종보고서

## Subscription Sharing Service

제출일: 2021년 12월 12일

소프트웨어 공학 10조

201900810 김성중

201501076 노재경

201702363 윤태현

201702436 이동규

201603840 황정연

## 목차

I. 시스템 개요.....	
1. 프로젝트명 .....	
2. 프로젝트 개요.....	
3. 프로젝트 산출물.....	
II. 참여인원별 산출물 및 기여비율 .....	
1. 작성 산출물 목록 .....	
2. 각 팀원이 구현한 Source Code의 MH .....	
3. 보고서 발표 및 회의록 작성자.....	
III. 테스트 .....	
1. JUNIT .....	
2. 테스트케이스 표 .....	
3. 테스트 결과 스크린샷 .....	
IV. 첨부 .....	

## I. 시스템 개요

### 1. 프로젝트명 - SSS

본 프로젝트의 이름인 SSS는 Subscription Sharing Service의 약자이다. SSS의 1차적 의미는 해당 서비스의 1차적 목적인 구독 서비스를 공유하는 플랫폼이라는 것이고, 2차적 의미는 구독을 통해 서로의 소통을 나누는 플랫폼이라는 것이다.

### 2. 프로젝트 개요

최근 대한민국의 가장 큰 위기는 코로나 바이러스로 인한 팬데믹이 아닐지도 모른다. 2021년 9월 28일 MBC 뉴스에서 <sup>1</sup>“우려스러운 건 40대 이상의 자살률은 모두 줄었는데 30대 이하, 특히 1·20대 청년들의 자살률이 크게 높아졌다는 점입니다. 10대 자살률은 2019년 10만 명당 5.9명에서 6.5명으로 9% 이상 늘었고, 20대 자살률은 19.2명에서 21.7명으로 12.8% 급증했습니다. 특히 10대 남성 자살률은 10만 명당 5.5명에서 6.5명으로 18.8% 증가, 20대 여성도 10만 명당 16.6명에서 19.3명으로 16.5% 증가를 기록했습니다.” 라는 보도를 내며 대한민국의 높은 청년 자살률이 현재의 대한민국이 마주한 큰 위기라고 시사했다.

전문가들은 소위 “코로나 블루”라고 불리는 ‘사회적 거리두기’ 정책의 장기화로 인한 정서적 고립이 이러한 위기의 큰 요인이라고 주장한다. 따라서 오늘날과 같은 팬데믹 상황에서는 반드시 정서적 고립을 막기 위한 소통이 이루어져야 하며, 현실적인 상황을 고려할 때 소통을 위한 창구로 가장 적절한 것은 온라인 커뮤니티와 같은 비대면 플랫폼일 것이다. 즉, 건강한 온라인 커뮤니티의 중요성이 지금만큼 중요한 시기는 없다고 봐도 무방하다.

온라인 커뮤니티가 건강하게 유지되고, 그 순기능을 극대화하기 위한 가장 좋은 기준은 ‘최대한 많은 사람들, 좋은 콘텐츠 아래에 모이게 하는 것’이다. 대표적인 경우로 방탄소년단의 팬덤인 “아미”를 들 수 있다. 방탄소년단의 수 많은 팬들은 물리적으로 함께 있을 수는 없지만, “BTS”라는 공동된 콘텐츠 아래 전세계에서 온라인 커뮤니티를 활용하여 기부와 자선행사 등 사회적으로 선한 영향력을 보이고 있다. 다시 말해, 좋은 콘텐츠에 동반되는 콘텐츠 이용자들 간의 건전한 커뮤니티 형성은 누군가와 함께 한다는 유대감을 통해 코로나 블루로 인한 우울감이나 외로움 등을 해소함은 물론 공익적인 활동으로도 이어질 수 있다는 것이다. 이렇게 생각해 보았을 때 전세계적인 흥행을 일으킨 넷플릭스 오리지널 시리즈 <오징어 게임>은 또 다른 콘텐츠의 예시가 될 수 있다. 해당 콘텐츠에 열광한 사람들은 각자의 소셜 미디어 계정 및 유튜브 영상의 댓글 등을 이용하여 드라마에 나온 내용들에 대한 자신들의 의견을 서로 활발히 나누었다.

이는 앞서 말한 것처럼 콘텐츠를 이용해 이용자들이 각자의 사회적 고립을 해결하는 모습들이다. 단, 현재는 이러한 소통의 중심점이 없고 자신과 대화를 한 사람과의 지속적인 의견 교류가 어렵다. 다시 말해, 현재의 열풍은 지속가능한 사회적 고립 해소로 이

---

<sup>1</sup> 이정은, “10 대 20 대가 위험하다..청년 자살률 왜 크게 늘었을까?“, MBC, 2021.09.28

어지지 않는다. 넷플릭스와 같은 OTT 콘텐츠에 대한 수요와 이를 매개로 하여 세상과 대화하고자 하는 사람들은 많으나 그것이 올바르게 이루어지지 않고 있는 것이다.

전 세계적으로 OTT서비스의 대표격인 넷플릭스 뿐만 아니라 최근에는 “디즈니 플러스”, “왓차” 등 다양한 OTT 서비스의 출현이 이어지고 있다. 대한민국 OTT서비스 시장 규모 역시 2020년을 기점으로 커지고 있는 추세이다. 특히 한국에서 만들어내는 k-콘텐츠는 넷플릭스로부터 2021년 한 해에만 <sup>2</sup>5500억원을 투자 받는 등 그 가치를 인정받고 있다. 이렇듯 국산 콘텐츠의 질과 양이 OTT서비스의 도움으로 비약적으로 발전하고 있는 상황에서, 이제는 한국인들이 이를 가지고 소통할 튼튼한 플랫폼이 필요하다.

### 3. 프로젝트 산출물

SSS는 웹기반의 커뮤니티로 두 가지 주요 기능을 제공한다.

1. 이용자 간 OTT 서비스 계정 공유 플랫폼
2. OTT 콘텐츠를 위한 소통 플랫폼

이번 프로젝트는 커뮤니티의 접근성과 훌륭한 사용자 경험을 위해 UI/UX에 많은 역량을 집중한다.

---

<sup>2</sup> 김예랑, “넷플릭스, 2021년 5500억 투자...韓오리지널 영화도 제작” <한국경제신문>, 2021.02.25

## II. 참여인원별 산출물 및 기여비율

### 1. 작성 산출물 목록

성명	작성 산출물
김성중	<ul style="list-style-type: none"> <li>- 개발 계획서</li> <li>1. UI 디자인</li> <li>- 요구 분석 명세서</li> <li>1. UI 프로토타입 제작</li> <li>2. 스토리보드 이미지 에셋 생성</li> <li>- 시스템 설계 명세서</li> <li>1. 테스트 요구사항 작성</li> <li>2. UI별 명세</li> <li>- 최종 보고서</li> <li>1. 테스트 케이스 작성(프론트엔드)</li> </ul>
노재경	<ul style="list-style-type: none"> <li>- 개발 계획서</li> <li>1. 위험 분석 작성</li> <li>2. 발표자료 준비</li> <li>- 요구 분석 명세서</li> <li>1. 유스케이스 다이어그램 작성</li> <li>2. 유스케이스 명세 작성</li> <li>- 시스템 설계 명세서</li> <li>1. 소개 작성</li> <li>2. 시퀀스 다이어그램 작성</li> <li>3. Data 설계 작성</li> </ul>
윤태현	<ul style="list-style-type: none"> <li>- 개발계획서</li> <li>1. 프로젝트 규모 - 자원 및 일정 예측</li> <li>2. CPM/간트차트</li> <li>- 요구 분석 명세서</li> <li>1. 기능적 요구사항 - 가. 상세 기능 요구사항</li> <li>2. 유스케이스 명세               <ul style="list-style-type: none"> <li>1) 액터 명세</li> <li>2) 유스케이스 명칭</li> <li>3) 유스케이스 명세(회원, 비회원)</li> </ul> </li> <li>- 시스템 설계 명세서</li> <li>1. 입출력 설계(관리자 UI-B 부분)</li> <li>2. Project Planning(Tasks, CPM, Task Allocation)</li> <li>3. UI별 명세</li> <li>- 최종보고서</li> <li>1. 시스템 개요</li> <li>2. 첨부문서 정리</li> </ul>
이동규	<ul style="list-style-type: none"> <li>- 개발 계획서</li> <li>1. 테스트 요구사항 작성</li> <li>2. 유지보수 요구사항 작성</li> <li>- 요구 분석 명세서</li> <li>1. 클래스 다이어그램 설계</li> <li>2. ppt 제작 및 발표</li> </ul>

	- 시스템 설계 명세서 1. 클래스 다이어그램 확장 2. MemberField & Method 명세 3. Data-flow Model 설계 4. 모듈 설계 - 최종 보고서 1. 단위 테스트 코드 작성(백엔드)
황정연	- 개발계획서 1. 프로젝트 개요 및 속성 설정 2. 조직도 및 인력 배치 3. 개발 절차 설정 - 요구 분석 명세서 1. 목적, 범위, 한계 설정 2. 설계 및 구현, 법적 제약사항 명시 - 시스템 설계 명세서 1. 클래스 다이어그램 2. UI별 명세 - 최종보고서 1. 테스트 케이스(통합)

## 2. 각 팀원이 구현한 Source Code의 MH

성명	코딩	MH
김성중	프론트엔드(레이아웃)	41
노재경	DB 구축	34
윤태현	DB 설계	30
이동규	백엔드 설계 및 코딩	42
황정연	프론트엔드와 백엔드 통신 구현	34

## 3. 보고서 발표 및 회의록 작성자

성명	발표	회의록
김성중	최종 보고서	개발계획서
노재경	개발 계획서	시스템 설계 명세서
윤태현	시스템 설계 명세서	최종 보고서
이동규	요구 분석 명세서	시스템 설계 명세서
황정연	시스템 설계 명세서	요구 분석 명세서

### Ⅲ. 테스트

#### 1. JUNIT

자바 프로그래밍 언어용 유닛 테스트 프레임워크으로, 테스트 결과는 Test 클래스로 개발자에게 테스트 방법 및 클래스의 History 를 공유 가능하게 한다. 해당 프레임워크는 단정(assert) 메서드로 테스트 케이스의 수행 결과를 판별하며, 어노테이션으로 간결하게 지원한다.

JUNIT 은 프로그래밍에서 모든 함수와 메서드에 대한 테스트 케이스(Test case)를 작성하여 의도된 대로 잘 동작하는지 검증하는 절차를 가지며, 프로그램을 작은 단위로 쪼개어 각 단위가 정확하게 동작하는지 검사함으로써 프로그램의 안정성을 높였다. 또한 System.out.println()으로 하는 번거로운 디버깅이 필요없으며, 개발기간 중 대부분을 차지하는 디버깅 시간을 단축할 수 있다.

테스트 결과를 일일이 눈으로 확인할 필요 없이 Assert 메서드로 통과 여부를 알 수 있음

Ex) `assertThat(user.name).isEqualTo("김철수")`

Name 이 “김철수가 맞다면 파란불, 아니라면 실패값과 기대값의 차이를 보여주며 노란불, Exception 발생시에는 빨간불을 보여준다.

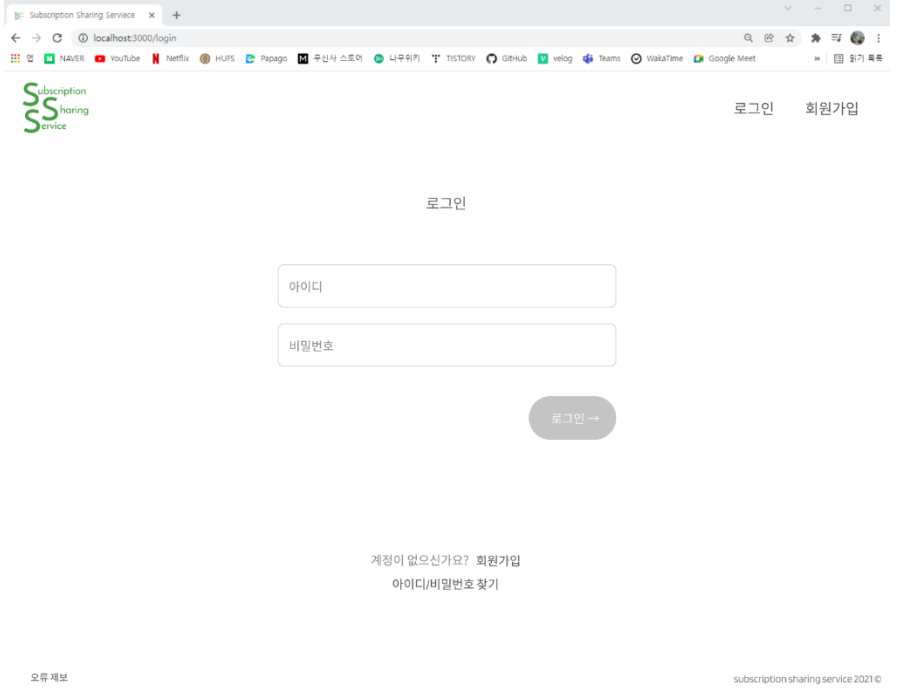
## 2. 테스트 케이스 표

순번	테스트 ID	테스트케이스 이름	예상 결과
1	1-1	일반 로그인	로그인 후 메인 페이지로 이동, Header 에는 마이페이지와 로그아웃이 표시됨
2	1-2	틀리게 기입한 로그인	로그인이 되지 않은 상태로 로그인 페이지에 남아 있음. 기입한 내용은 삭제됨
3	2-1	일반 회원가입	조건대로 기입하면 회원가입 성공. 로그인 된 메인페이지로 이동.
4	2-2	올바르지 않은 회원가입 시도	올바르지 않은 정보라고 소개 후 회원가입 페이지에 남아있음.
5	2-3	중복된 회원가입 시도	이미 있는 계정이라고 소개 후 회원가입 페이지에 남아 있음.
6	3-1	타인 게시글 조회	해당 게시글의 상세 페이지로 이동. 수정과 삭제는 불가능함.
7	3-2	게시글 작성	로그인 된 상태에서 현재 게시판에 게시글 작성
8	3-3	자기 게시글 조회	해당 게시글의 상세 페이지로 이동. 수정과 삭제 탭이 활성화됨.
9	3-3-1	자기 게시글 수정	기존 기입했던 내용이 첨부되어있는 채로 글쓰기 페이지로 이동. 수정을 하면 수정된 글이 저장됨.
10	3-3-2	자기 게시글 삭제	삭제를 원하는지 더블체크 실행. 그럼에도 삭제가 선택될 경우 글 삭제.
11	3-4	게시글 추천	게시물의 추천 버튼을 누르면 숫자가 1 올라가며 데이터베이스에 추천 숫자가 저장됨.
12	3-5	게시글 신고	게시물의 신고 버튼을 누르면 숫자가 1 올라가며 신고 숫자가 저장됨.
13	3-6	댓글 작성	게시글에 댓글을 작성
14	3-7	회원 마이페이지	회원정보 및 현재까지 업로드한 포스팅들을 조회할 수 있음.
15	4-1	관리자 게시글 조회	해당 게시글의 상세 페이지로 이동. 삭제가 가능함.

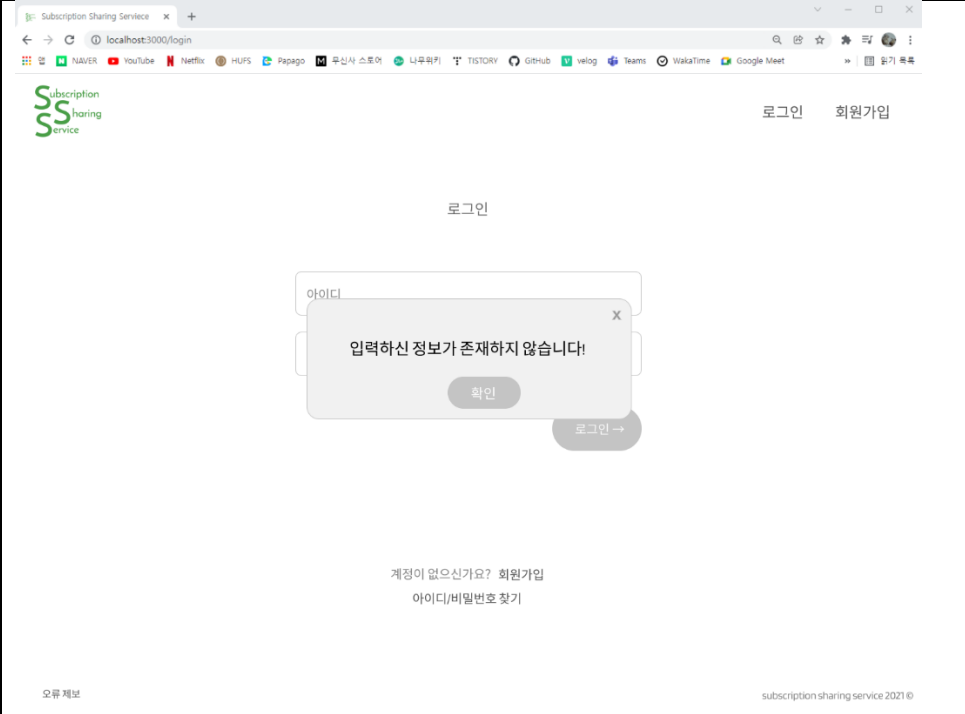


### 3. 테스트 결과 스크린샷

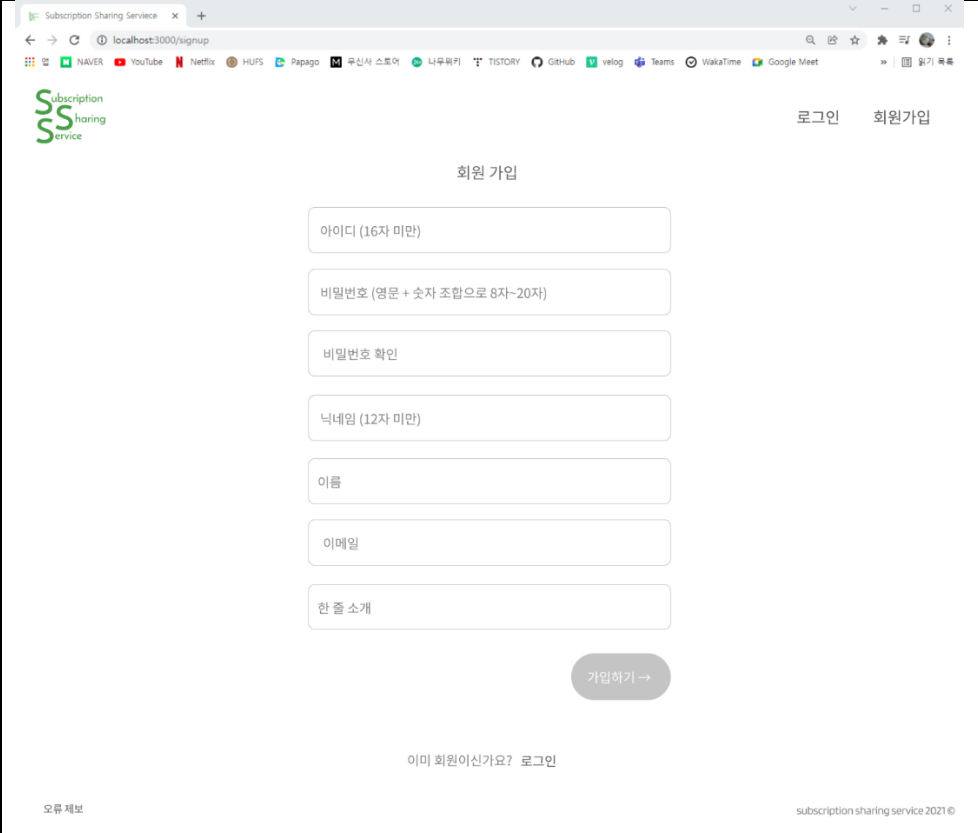
#### 1-1. 일반 로그인

클라이언트	
서버	<pre data-bbox="486 1061 1401 1487"> @Test public void login() throws Exception {     //given     User user = userService.join( loginId: "ldk", password: "1234", name: "이등규",         nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");      //when     User loginUser = loginService.login( loginId: "ldk", password: "1234").get();      //then     assertThat(user).isEqualTo(loginUser); }                     </pre>

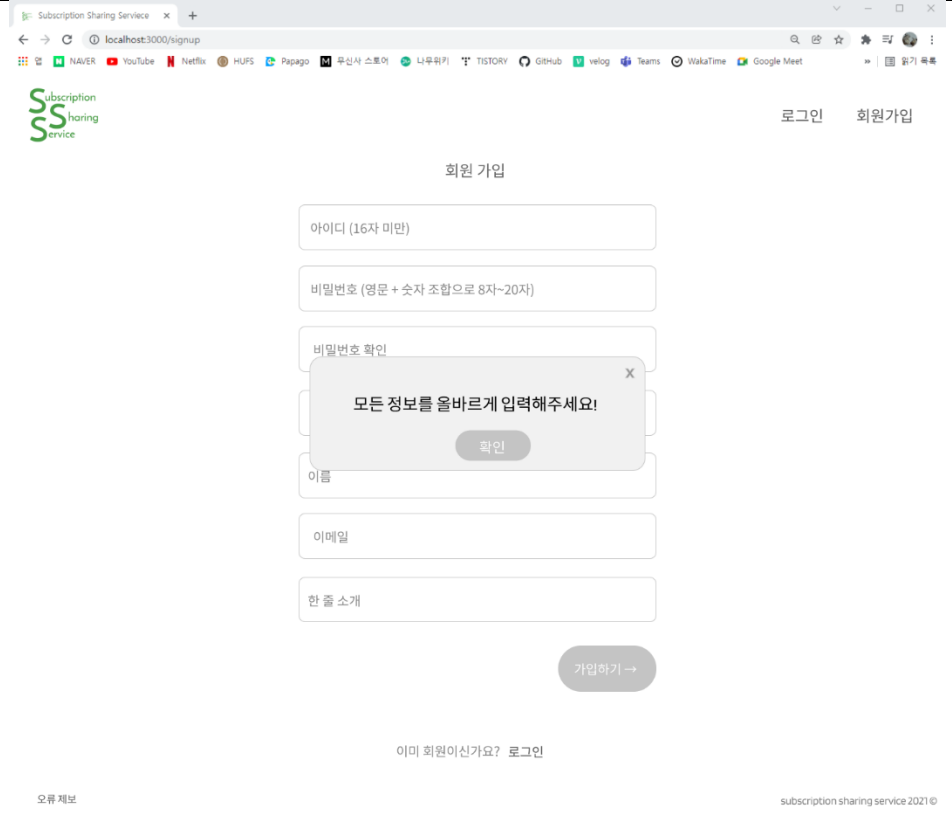
## 1-2. 틀리게 기입한 로그인

클라이언트	
서버	<pre data-bbox="432 1064 1401 1449"> @Test public void loginFail() throws Exception {     //given     User user = userService.join( loginId: "ldk", password: "1234", name: "이동규",         nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");      //when     Optional&lt;User&gt; findUser = loginService.login( loginId: "ldk", password: "123");      //then     assertThat(findUser).isEmpty(); }                     </pre>

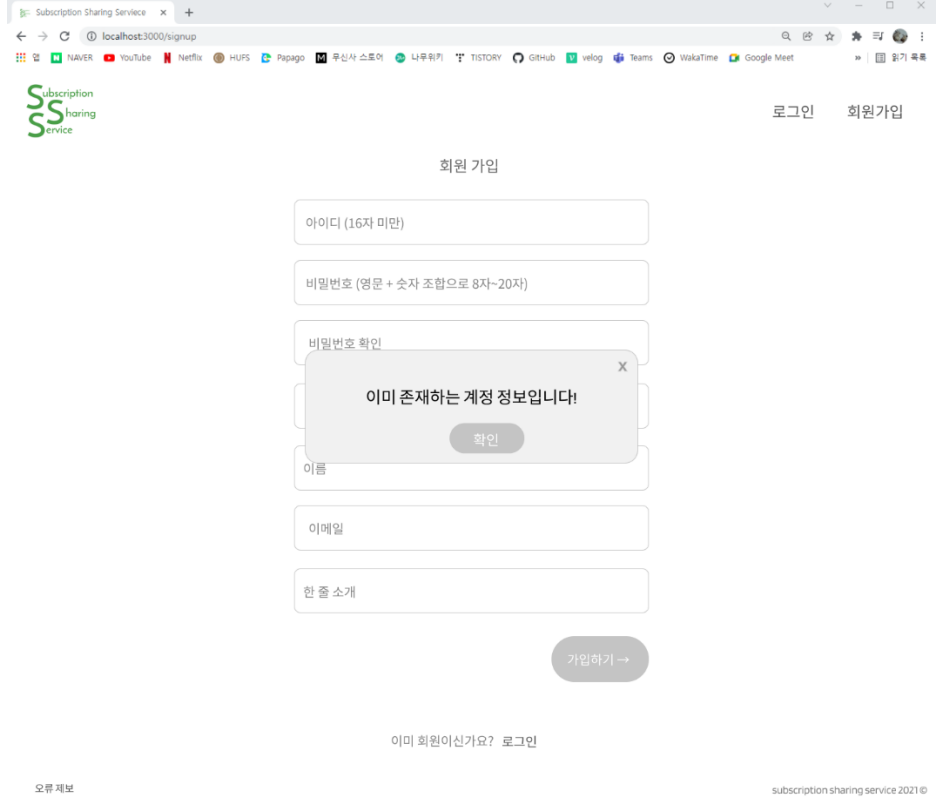
## 2-1. 일반 회원가입

클라이언트	
서버	<pre data-bbox="419 1182 1401 1505"> @Test public void join() throws Exception {     //given     //when     User user = userService.join( loginId: "ldk", password: "1234", name: "이동규",         nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");     //then     User findUser = userService.findById("ldk").get();     assertThat(user).isEqualTo(findUser); } </pre>

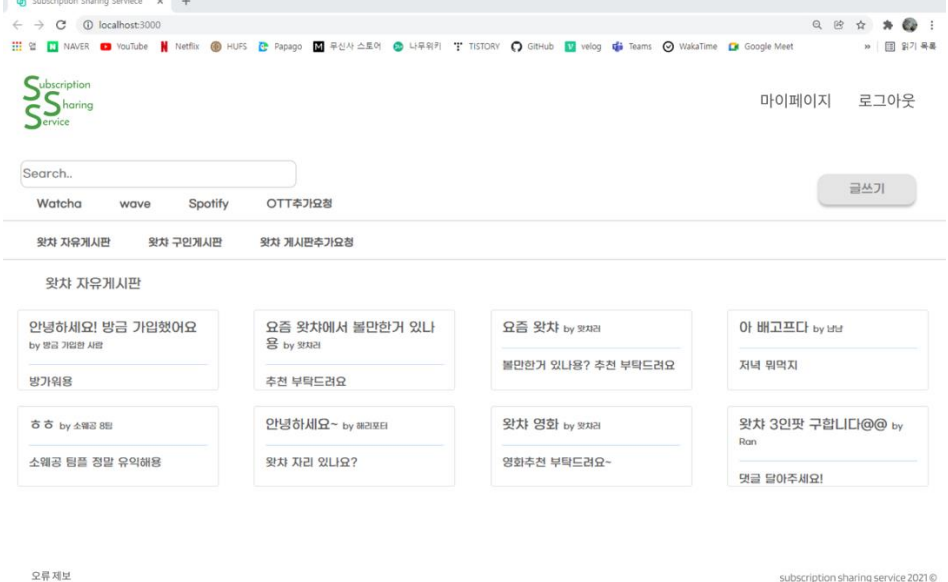
## 2-2. 올바르지 않은 회원가입 시도

클라이언트	
서버	<pre data-bbox="459 1182 1374 1588"> @Test(expected = IllegalStateException.class) public void validateLoginId() throws Exception {     //given      //when     User user1 = userService.join( loginId: "ldk", password: "1234", name: "이동규",                                    nickName: "dka", introduce: "안녕", email: "980130@gmail.com");     User user2 = userService.join( loginId: "ldk", password: "1234", name: "이동규",                                    nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");      //then     fail(); }                 </pre>

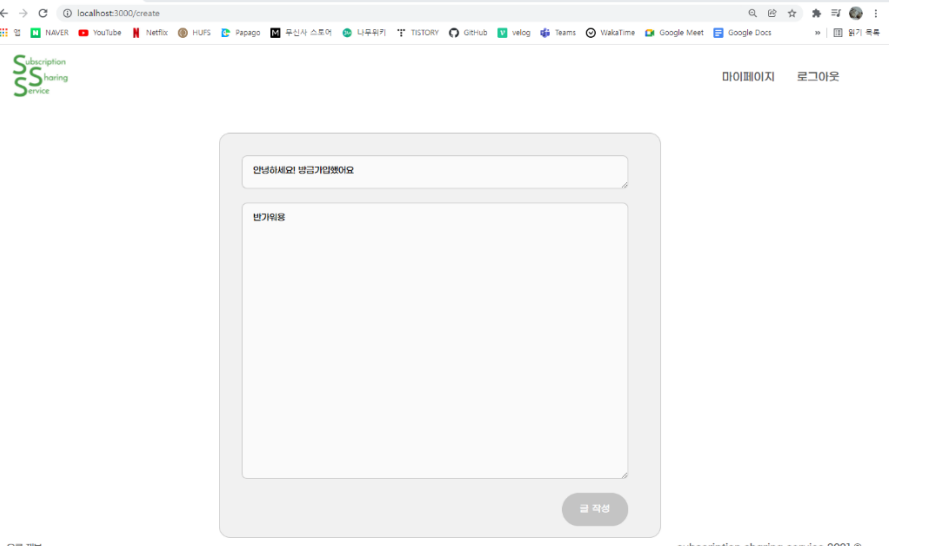
### 2-3. 중복된 회원가입 시도 (중복 아이디)

클라이언트	
서버	<pre data-bbox="469 1111 1374 1505"> @Test(expected = IllegalStateException.class) public void validateLoginId() throws Exception {     //given      //when     User user1 = userService.join( loginId: "ldk", password: "1234", name: "이동규",                                    nickName: "dka", introduce: "안녕", email: "980130@gmail.com");     User user2 = userService.join( loginId: "ldk", password: "1234", name: "이동규",                                    nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");      //then     fail(); }                     </pre>

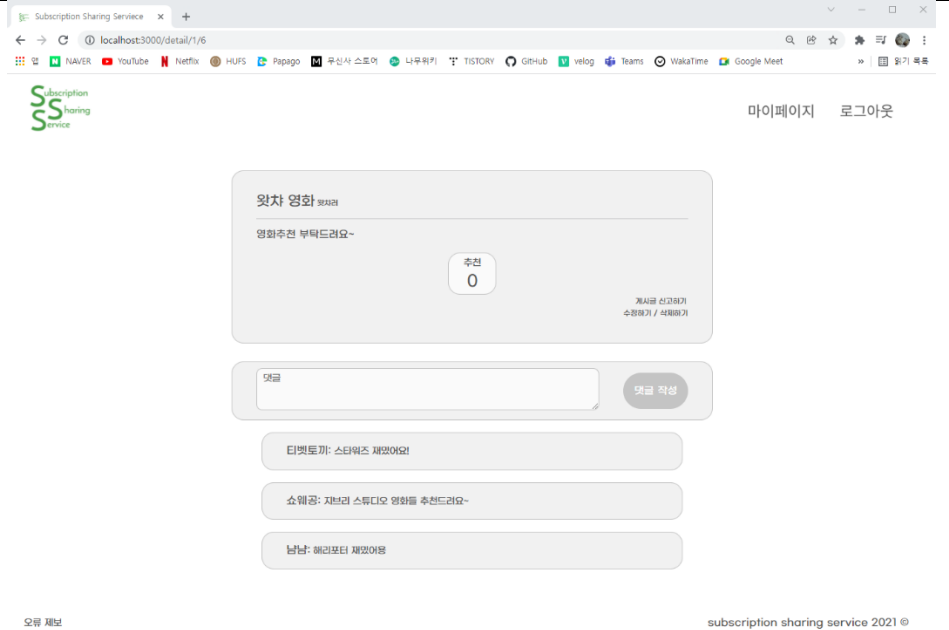
### 3-1. 타인 게시물 조회

클라이언트	
서버	<pre> @Test public void findOtherPost() throws Exception {     //given     Board board = boardRepository.save(Board.create("왓차"));     Category category = categoryRepository.save(Category.create("진목", board));     User user1 = userService.join(loginId: "ldk", password: "1234", name: "이동규",         nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");     Long postId = postService.register(user1.getId(), category.getId(), title: "제목", content: "내용").getId();     userService.join(loginId: "asd", password: "1234", name: "김철수",         nickName: "ch", introduce: "안녕", email: "chulsoo@gmail.com");      //when     Optional&lt;Post&gt; findPost = postService.findById(postId);      //then     assertThat(findPost.get().getTitle()).isEqualTo("제목");     </pre>

### 3-2. 게시글 작성

클라이언트	
서버	<pre> @Test public void postRegister() throws Exception {     //given     Board board = boardRepository.save(Board.create("왓차"));     Category category = categoryRepository.save(Category.create( name: "진목", board));     User user = userService.join( loginId: "ldk", password: "1234", name: "이동규",                                 nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");      //when     Post post = postService.register(user.getId(), category.getId(), title: "제목", content: "내용");      //then     Post findPost = postService.findById(post.getId()).get();     assertThat(findPost).isEqualTo(post); }         </pre>

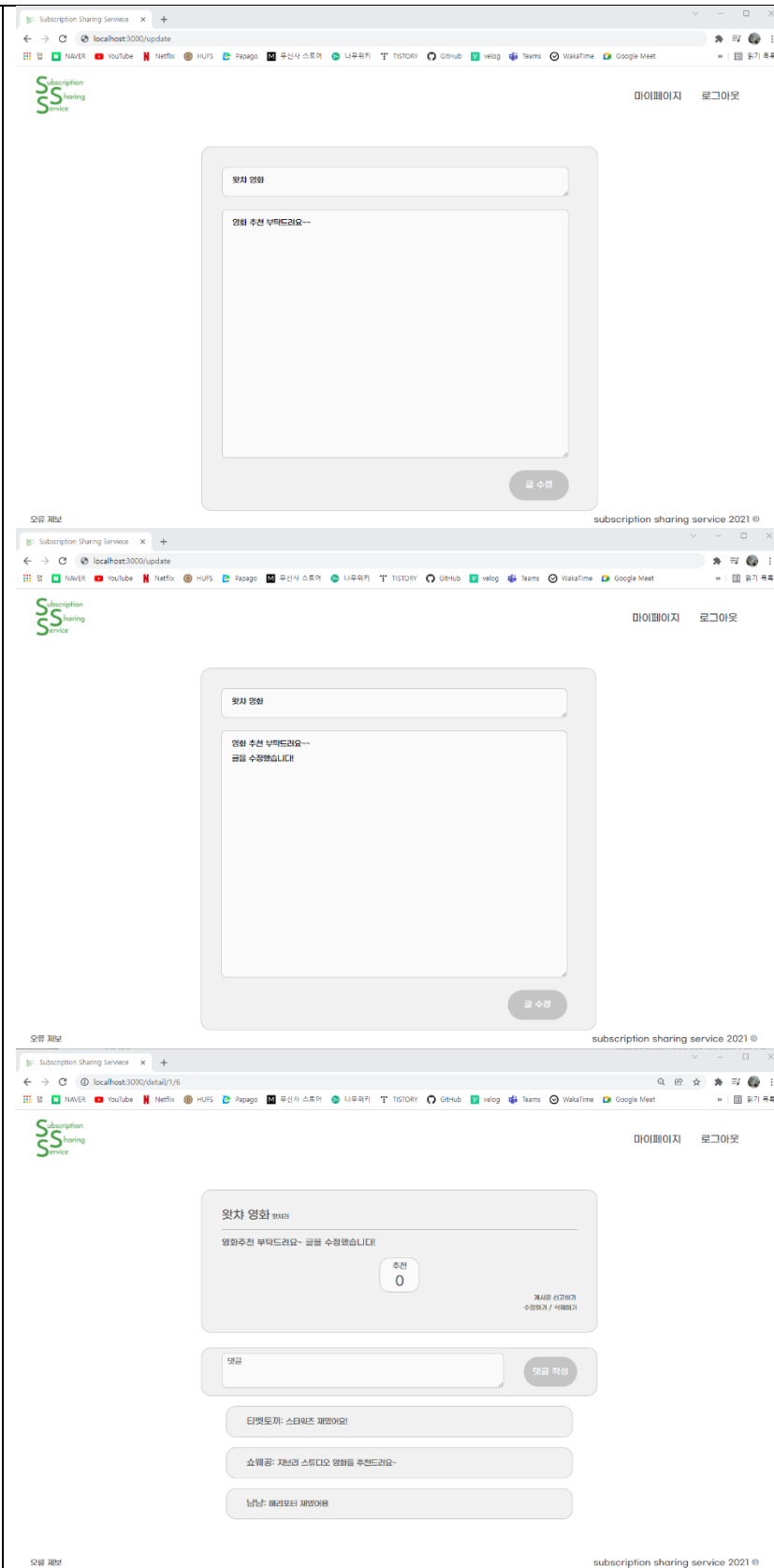
## 3-3. 자기 게시글 조회

클라이언트	
서버	<pre data-bbox="448 981 1401 1348"> @Test public void findMyPost() throws Exception {     //given     Board board = boardRepository.save(Board.create("찾자"));     Category category = categoryRepository.save(Category.create( name: "진록", board));     User user = userService.join( loginId: "ldk", password: "1234", name: "이동규",                                 nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");     Long postId = postService.register(user.getId(), category.getId(), title: "제목", content: "내용").getId();      //when     Optional&lt;Post&gt; post = postService.findById(postId);      //then     assertThat(post.get().getUser()).isEqualTo(user); } </pre>



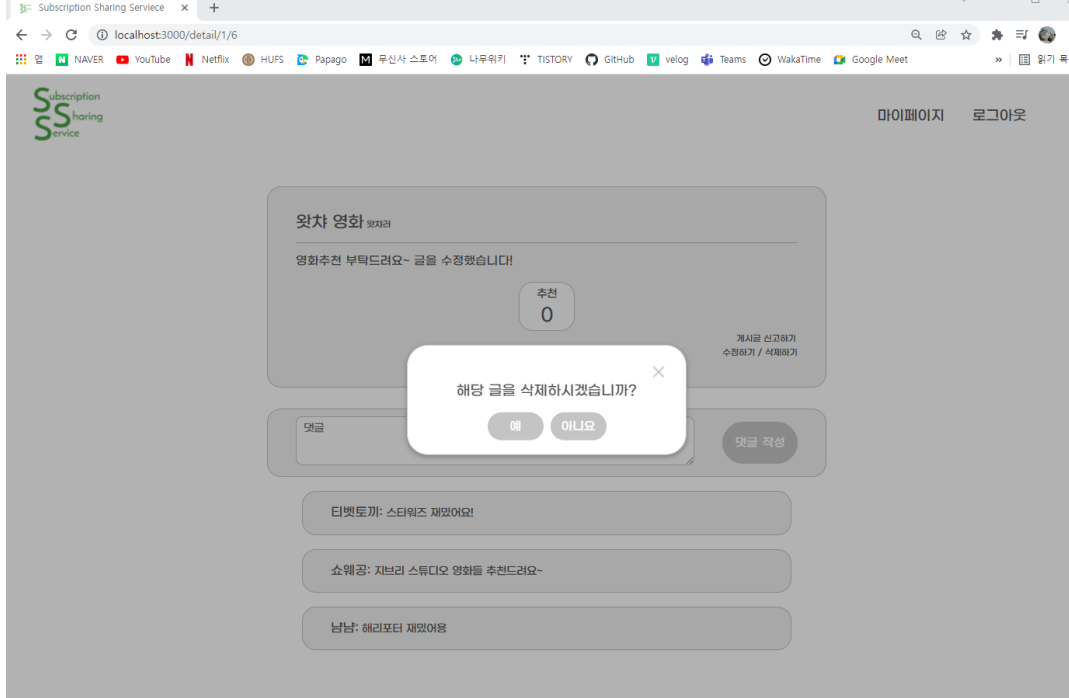
### 3-3-1. 자기 게시글 수정

클라이언트



<p>서버</p>	<pre> @Test public void postEdit() throws Exception {     //given     Board board = boardRepository.save(Board.create("왓차"));     Category category = categoryRepository.save(Category.create( name: "진목", board));     User user = userService.join( loginId: "ldk", password: "1234", name: "이동규",         nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");     Post post = postService.register(user.getId(), category.getId(), title: "제목", content: "내용");      //when     postService.edit(post.getId(), title: "바뀐 제목", content: "바뀐 내용");     String title = postService.findById(post.getId()).get().getTitle();     String content = postService.findById(post.getId()).get().getContent();      //then     assertThat(title).isEqualTo("바뀐 제목");     assertThat(content).isEqualTo("바뀐 내용"); }         </pre>
-----------	---

### 3-3-2. 자기 게시물 삭제

<p>클라이언트</p>	 <p>The screenshot shows a web browser window with the URL 'localhost:3000/detail/1/6'. The page displays a post titled '왓차 영화' (What's the deal with movie) with a recommendation count of 0. A modal dialog is open in the center, asking '해당 글을 삭제하시겠습니까?' (Do you want to delete this post?). The dialog has '예' (Yes) and '아니오' (No) buttons. The background page shows a list of movie recommendations: '디베토끼: 스타워즈 재밌어요!', '쇼웨공: 자브라 스튜디오 영화들 추천드려요~', and '남남: 해리포터 재밌어용'. The footer includes '오류 제보' and 'subscription sharing service 2021 ©'.</p>
--------------	--

서버

```

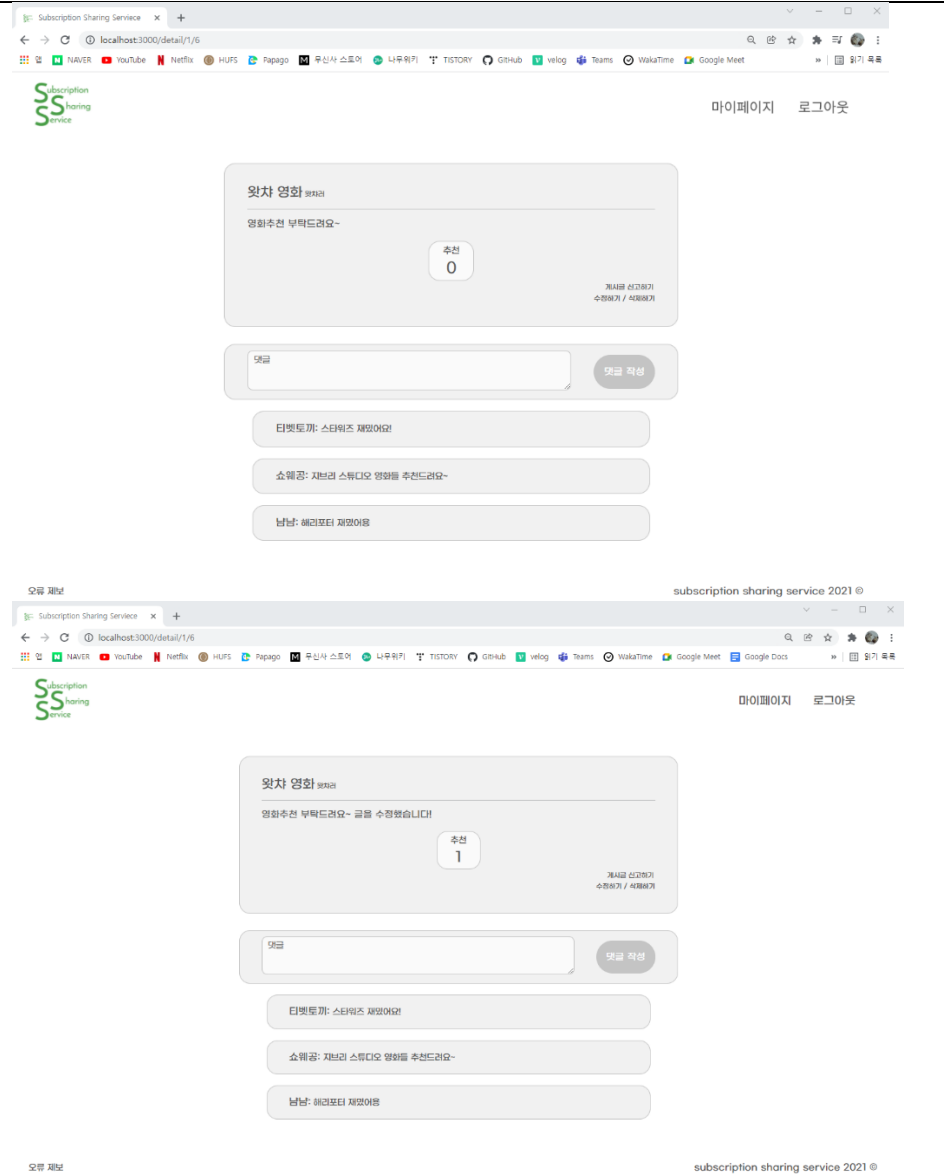
@Test
public void postDelete() throws Exception {
    //given
    Board board = boardRepository.save(Board.create("왓차"));
    Category category = categoryRepository.save(Category.create(name: "진목", board));
    User user = userService.join(loginId: "ldk", password: "1234", name: "이동규",
        nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");
    Post post = postService.register(user.getId(), category.getId(), title: "제목", content: "내용");
    Long postId = post.getId();

    //when
    int categoryPosts = category.getPosts().size();
    postService.delete(postId);
    Optional<Post> findPost = postService.findById(postId);

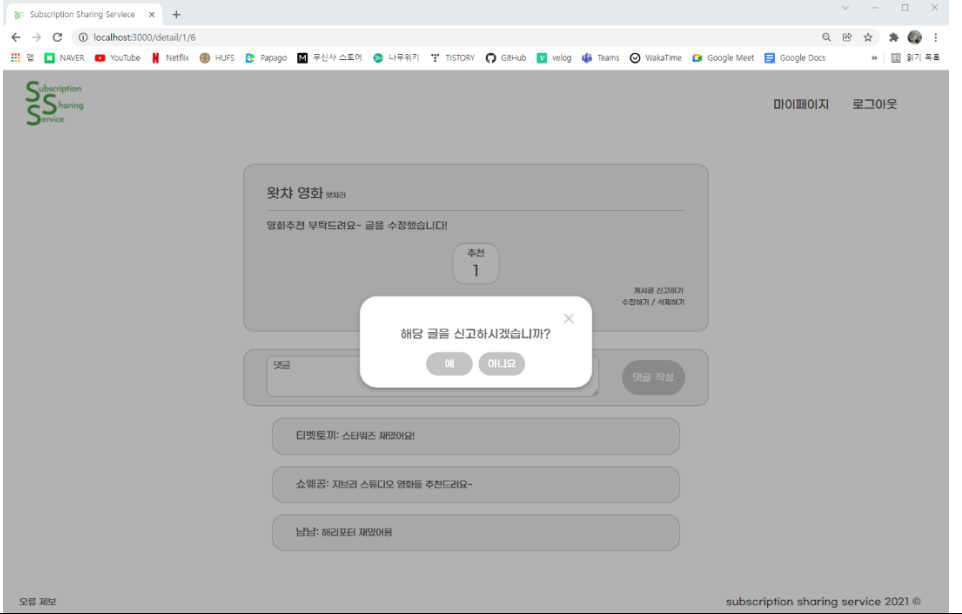
    //then
    assertThat(findPost).isEmpty();
    assertThat(category.getPosts().size()).isEqualTo(categoryPosts - 1);
}

```

### 3-4. 게시글 추천

<p>클라이언트</p>	
<p>서버</p>	<pre data-bbox="453 1458 1401 1906"> @Test public void recommend() throws Exception {     //given     Board board = boardRepository.save(Board.create("찾아"));     Category category = categoryRepository.save(Category.create( name: "진목", board));     User user = userService.join( loginId: "ldk", password: "1234", name: "이동규",         nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");     Post post = postService.register(user.getId(), category.getId(), title: "제목", content: "내용");      //when     int beforeSize = post.getRecommends().size();     recommendService.recommend(user.getId(), post.getId());     int afterSize = post.getRecommends().size();      //then     assertThat( actual: beforeSize + 1).isEqualTo(afterSize); }                     </pre>

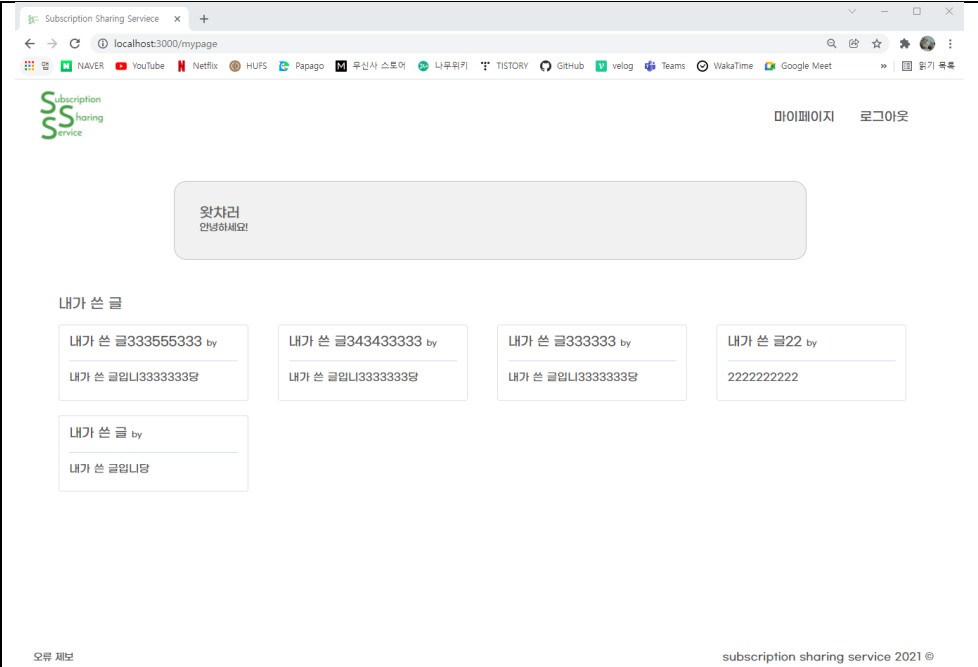
### 3-5. 게시글 신고

<p>클라이언트</p>	
<p>서버</p>	<pre> @Test public void reportPost() throws Exception {     //given     Board board = boardRepository.save(Board.create("왓차"));     Category category = categoryRepository.save(Category.create( name: "진목", board));      User user1 = userService.join( loginId: "ldk", password: "1234", name: "이동규",                                    nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");     User user2 = userService.join( loginId: "ldk1", password: "12345", name: "이동규",                                    nickName: "dk1", introduce: "안녕", email: "ldk@gmail.com");      Post post = postService.register(user1.getId(), category.getId(), title: "제목", content: "내용");      //when     int beforeSize = reportService.findAll().size();     Report report = reportService.reportPost(user1.getId(), post.getId(), Reason.CURSE);     reportService.reportPost(user2.getId(), post.getId(), Reason.FALSE);     int afterSize = reportService.findAll().size();      //then     assertThat( actual: beforeSize + 1).isEqualTo(afterSize);     assertThat(report.getInfos().size()).isEqualTo(2); }         </pre>

### 3-6. 댓글 작성

<p>클라이언트</p>	
<p>서버</p>	<pre> @Test public void commentRegister() throws Exception {     //given     Board board = boardRepository.save(Board.create("왓차"));     Category category = categoryRepository.save(Category.create(name: "전목", board));     User user = userService.join(loginId: "ldk", password: "1234", name: "이동규",         nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");     Post post = postService.register(user.getId(), category.getId(), title: "제목", content: "내용");      //when     Comment comment = commentService.register(user.getId(), post.getId(), content: "댓글1");      //then     Comment findComment = commentService.findById(comment.getId()).get();     assertThat(findComment.getContent()).isEqualTo("댓글1"); }         </pre>

### 3-7. 회원 마이 페이지

클라이언트	
서버	<pre data-bbox="419 952 1401 1489"> @Test public void myPage() throws Exception {     //given     Board board = boardRepository.save(Board.create("왓차"));     Category category = categoryRepository.save(Category.create( name: "진목", board));     User user = userService.join( loginId: "ldk", password: "1234", name: "이동규",         nickName: "dk", introduce: "안녕", email: "ldk980130@gmail.com");     postService.register(user.getId(), category.getId(), title: "제목", content: "내용");     postService.register(user.getId(), category.getId(), title: "제목", content: "내용");     postService.register(user.getId(), category.getId(), title: "제목", content: "내용");     postService.register(user.getId(), category.getId(), title: "제목", content: "내용");      //when     MyPage myPage = user.toMyPage();      //then     assertThat(myPage.getPosts().size()).isEqualTo(4); } </pre>

## IV. 첨부

번호	첨부 파일명
1	10조_01_개발방법론.pdf
2	10조_02_프로젝트 개발 계획서.pdf
3	10조_03_요구분석 명세서.pdf
4	10조_04_Use Case Diagram.pdf
5	10조_05_시스템 설계 명세서.pdf
6	10조_06_Data 설계.pdf
7	10조_07_입출력 설계.pdf
8	10조_08_Source Code.pdf