

1-1. 개발 목표

- **개발 목표**: 조직 구성원 간 원활한 의사소통을 위한 메신저 개발
- **개발 범위**: 채팅 서버 및 클라이언트
- **사용자 계층**: 포스코 ICT 전직원
- **사용자 인터페이스**: Web
- **과제 요구 사항**
 - 협업 기능(메시지 포스팅, 사진 및 문서 공유, 일대일, 다자간 메시지 통신) 등 기본 기능 중심으로 개발
 - 빠른 채팅과 음성/영상 처리가 가능한 모바일 최적화 메신저 앱 개발 등 향후 개발 방안을 제시
 - 메신저는 직원들끼리 사내에서 사용하므로 암호화나 보안 기능 제시
 - 모든 클라이언트에 실시간 동기화되는 특징 활용 모바일 환경에 채팅앱 개발을 위한 리얼타임 데이터베이스 구현

1-2. 개발 환경

구분	상세 내용
OS	Window10
Front-end	React / Redux / JS / sementic-ui-react
Back-end	Firebase Realtime Database / Firebase Storage / Firebase Authentication
Tool	IntelliJ Ultimate / Visual Studio Code


2-1. 주요 기능 및 구조 설계


주요기능	상세내용	진행상태
유저 가입 및 로그인	<ul style="list-style-type: none"> • 사용자 가입 및 로그인처리 • 이메일 인증 및 연동 로그인 	완료
유저 데이터 저장	<ul style="list-style-type: none"> • 사용자 정보를 저장하고 관리 	완료
유저 리스팅 화면	<ul style="list-style-type: none"> • 사용자 정보를 읽어와 사용자 목록 화면 구현 	완료
채팅 화면 & 채팅 메시지 리스팅	<ul style="list-style-type: none"> • 사용자 이름 클릭시 사용자와 Direct Message 기능 제공 • 채널 이름 클릭시 해당 채널 사용자와 그룹 채팅 기능 제공 	완료
채팅 메시지 전송	<ul style="list-style-type: none"> • 채팅방 화면에서 메시지를 입력하고 엔터를 누르거나 전송 버튼을 누르면 메시지가 전송 되는 기능 	완료
채팅방 리스팅	<ul style="list-style-type: none"> • 권한에 맞는 채팅방 리스팅 	완료
채팅방 초대	<ul style="list-style-type: none"> • Private Channel일 경우 다른 유저를 초대하는 기능 	완료
접속중인 유저 표시	<ul style="list-style-type: none"> • 회원목록에서 회원의 접속상태 확인 • 채팅방 초대시 회원의 접속상태 확인 	완료
파일 전송 기능(선택)	<ul style="list-style-type: none"> • 이미지 전송 기능 제공 	완료
데이터 베이스 (선택)	<ul style="list-style-type: none"> • 모든 클라이언트에 실시간 동기화되는 특징 활용하기 위한 리얼타임 데이터베이스 구현 	완료


2-2. 추가기능 및 구조 설계

주요기능	상세내용	진행상태
채팅방 정보	<ul style="list-style-type: none"> • 채팅방을 만든 사람 정보 제공 • Top Poster 정보 제공 • 채팅방의 상세정보 제공 • 채팅방의 멤버정보 제공 • 채팅 검색기능 제공 	완료
채팅방 타입	<ul style="list-style-type: none"> • 4가지 종류의 채팅방 타입 제공 (Direct, Public, Private, Secure) • Direct - 개인간의 대화 • Public - 모든 사용자가 접속 가능한 오픈 채팅방 • Private - 초대받은 멤버만 접속 가능한 채팅방 • Secure - 비밀번호로 접속 가능한 채팅방 	완료
즐거찾기 기능	<ul style="list-style-type: none"> • 채팅방을 즐겨찾기 등록 가능 	완료
이모티콘 전송 기능	<ul style="list-style-type: none"> • 메시지 이모티콘 전송 기능 	완료
읽지않은 메시지 표시	<ul style="list-style-type: none"> • 읽지않은 메시지와 양 표시 기능 	완료
채팅시간 표시 기능	<ul style="list-style-type: none"> • 채팅시간 표시 	완료
프로필 사진 변경 기능	<ul style="list-style-type: none"> • 프로필 사진 변경 가능 	완료

3-1. Login & Register (유저 가입 및 로그인)


 Login for poslack!!


 이메일


 비밀번호


Login


Don't have an account? [Register](#)

 Register for poslack!!

 이름

 이메일

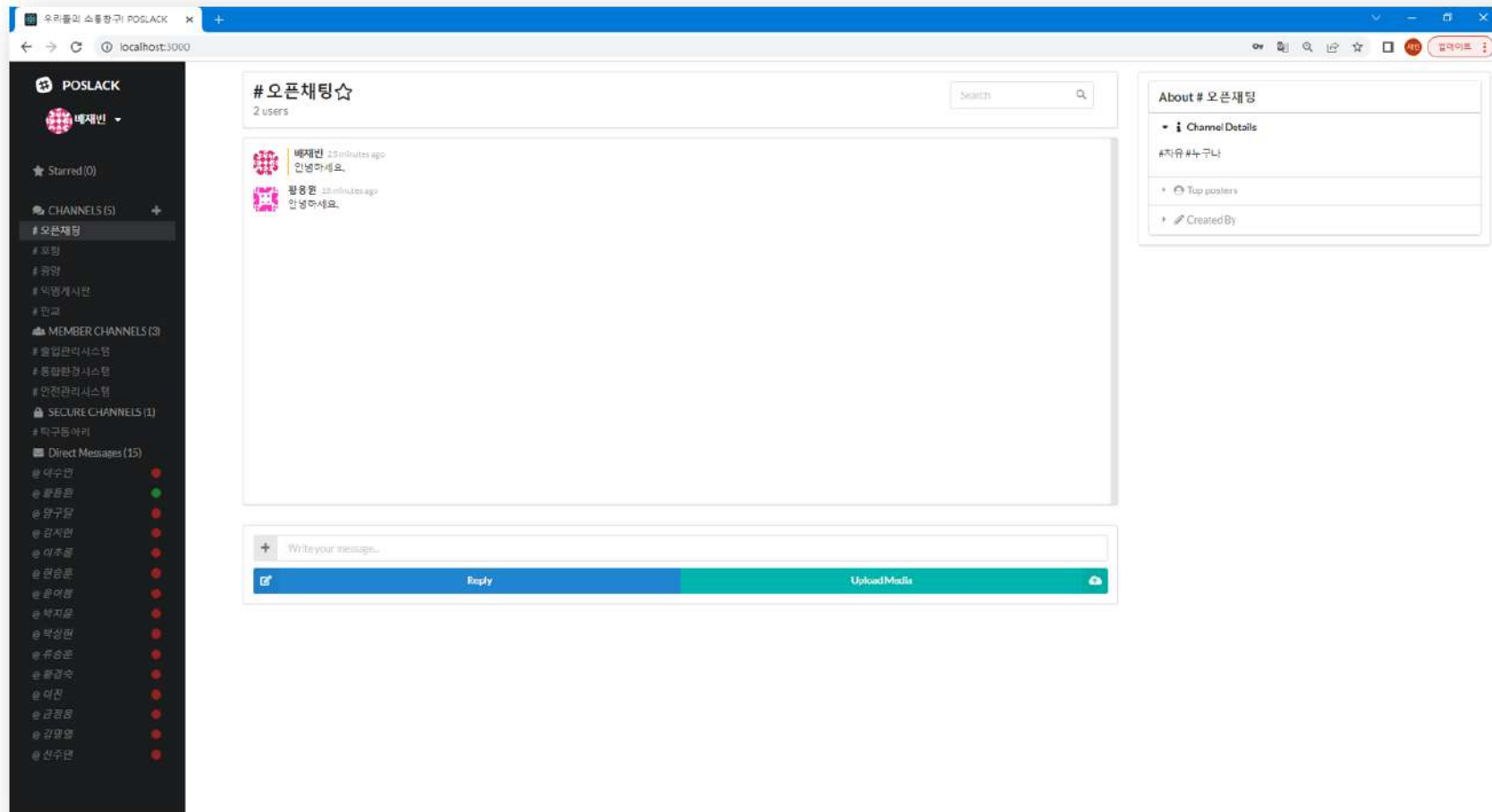
 비밀번호

 비밀번호 확인

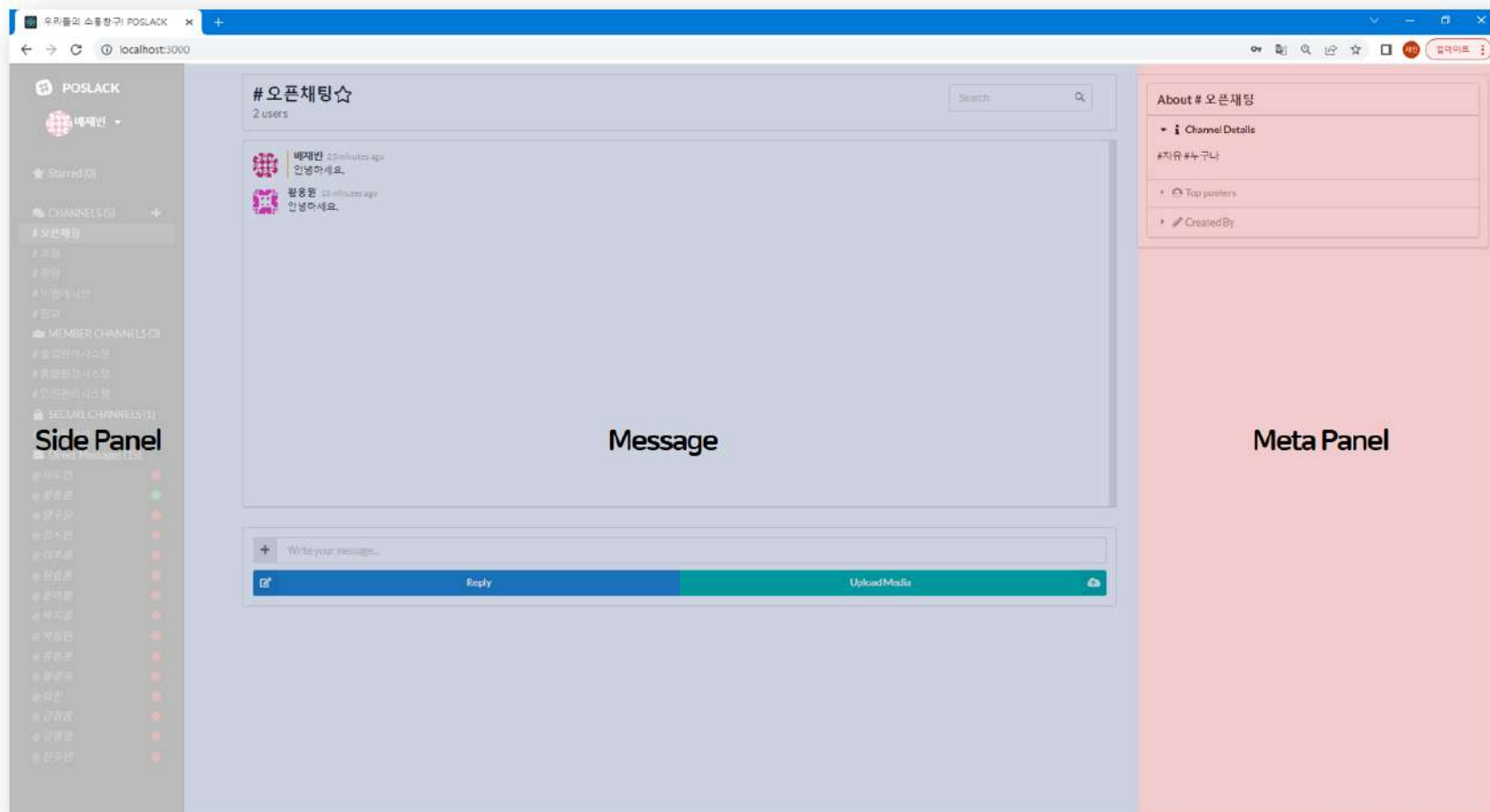
Submit

Already have an account? [Login](#)

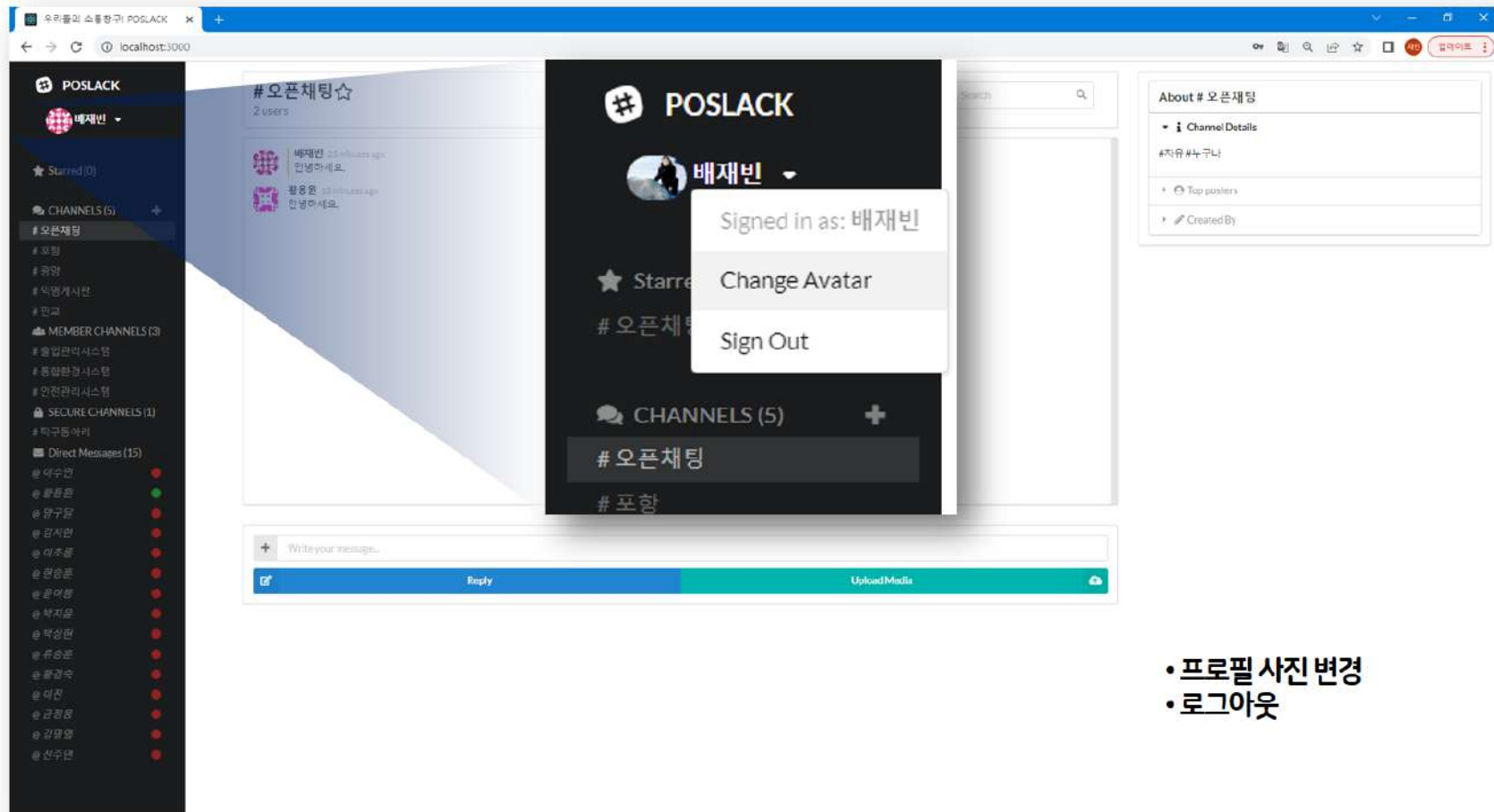
3-2. Main



3-2, Main



3-3. Side Panel > User Panel

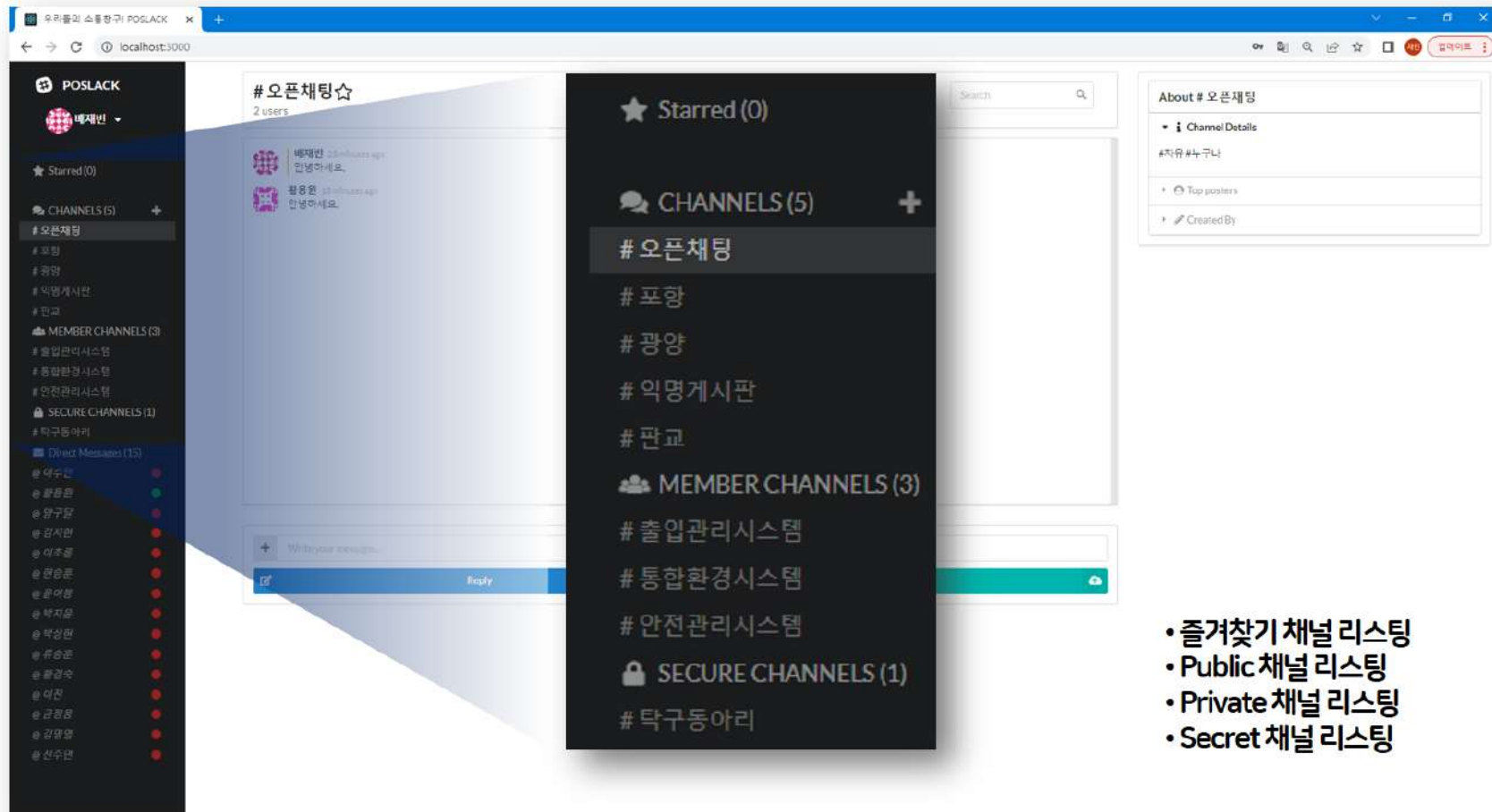


3-3. Side Panel > User Panel > Change Avatar (Modal)



- 프로필 사진 영역 설정
- 프로필 사진 미리 보기
- 프로필 사진 저장

3-3. Side Panel > Channels



3-3. Side Panel > Channels > Create Channel (Modal)

Create Channel

Name of Channel

이름

About the Channel

설명

About the Permission

Secret

비밀번호

비밀번호 확인

✓ Create

✗ Cancel

Enter Password

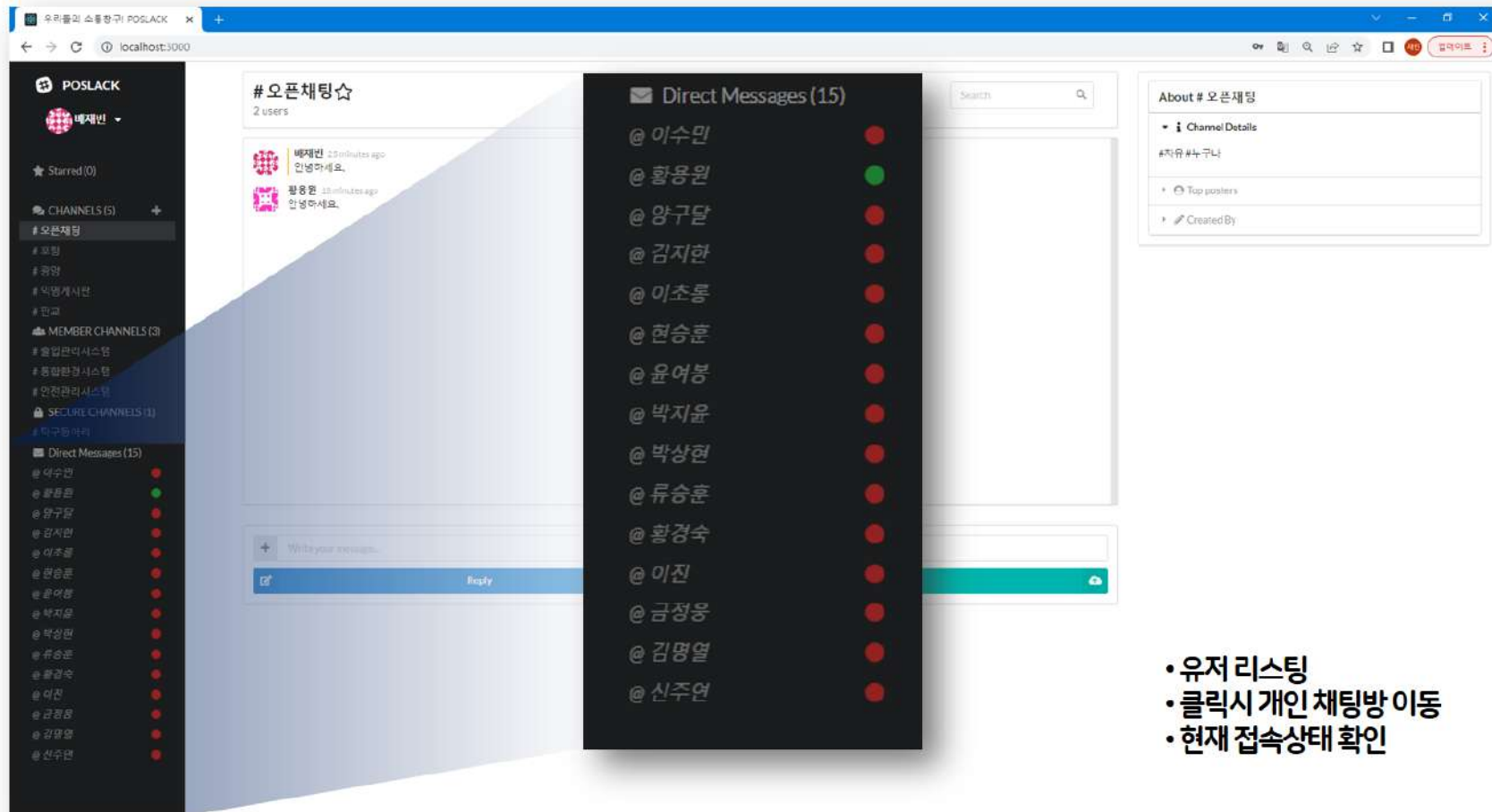
채널 비밀번호

✓ Accept

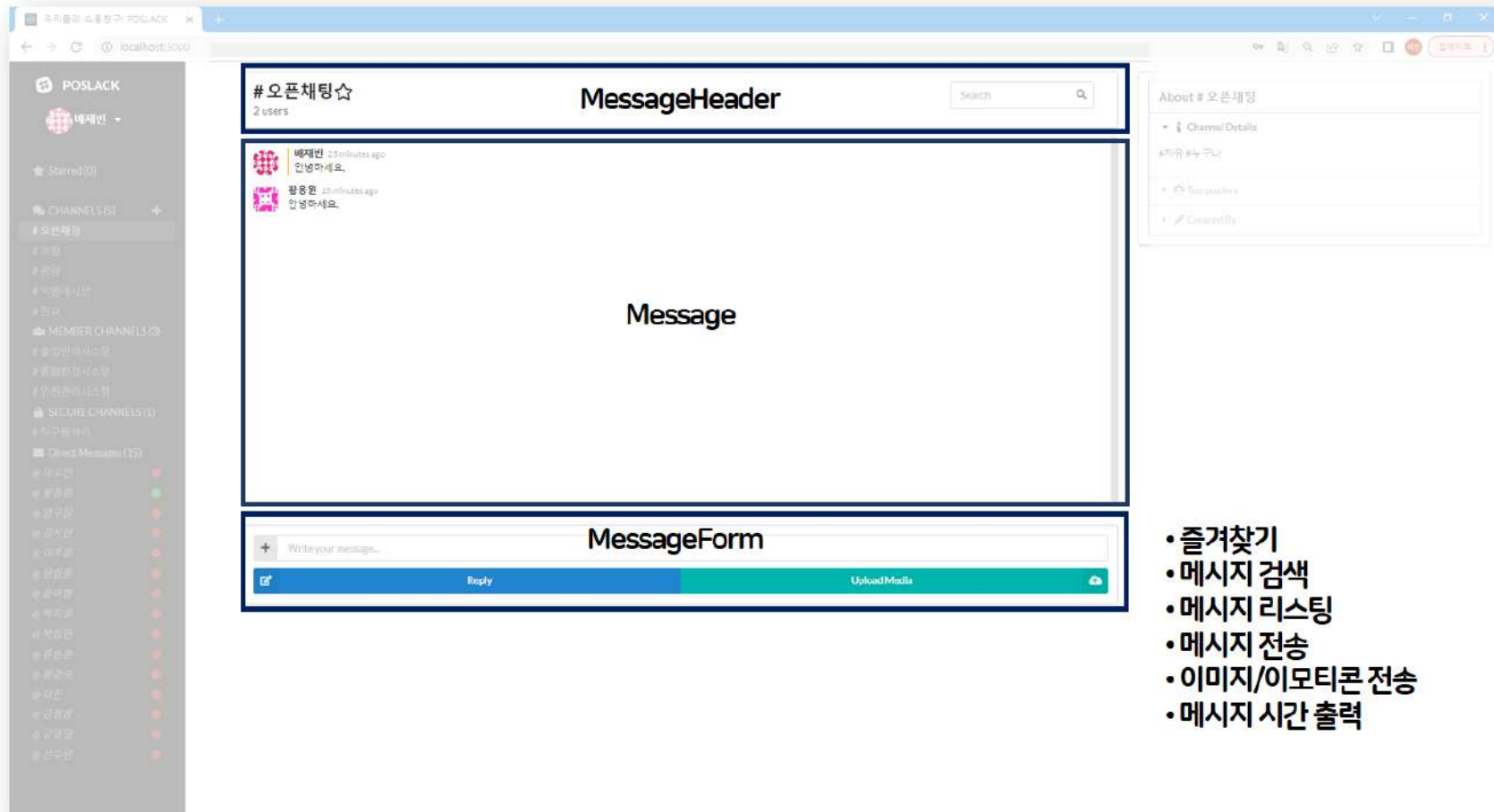
✗ Cancel

- 채널 생성
- 3가지 채널 유형 제공
- **Public**
누구나 입장 가능한 오픈채팅
- **Private**
멤버로 등록되어야 입장 가능
(멤버가 아니라면 리스팅X)
- **Secret**
채널 입장시 비밀번호 필요

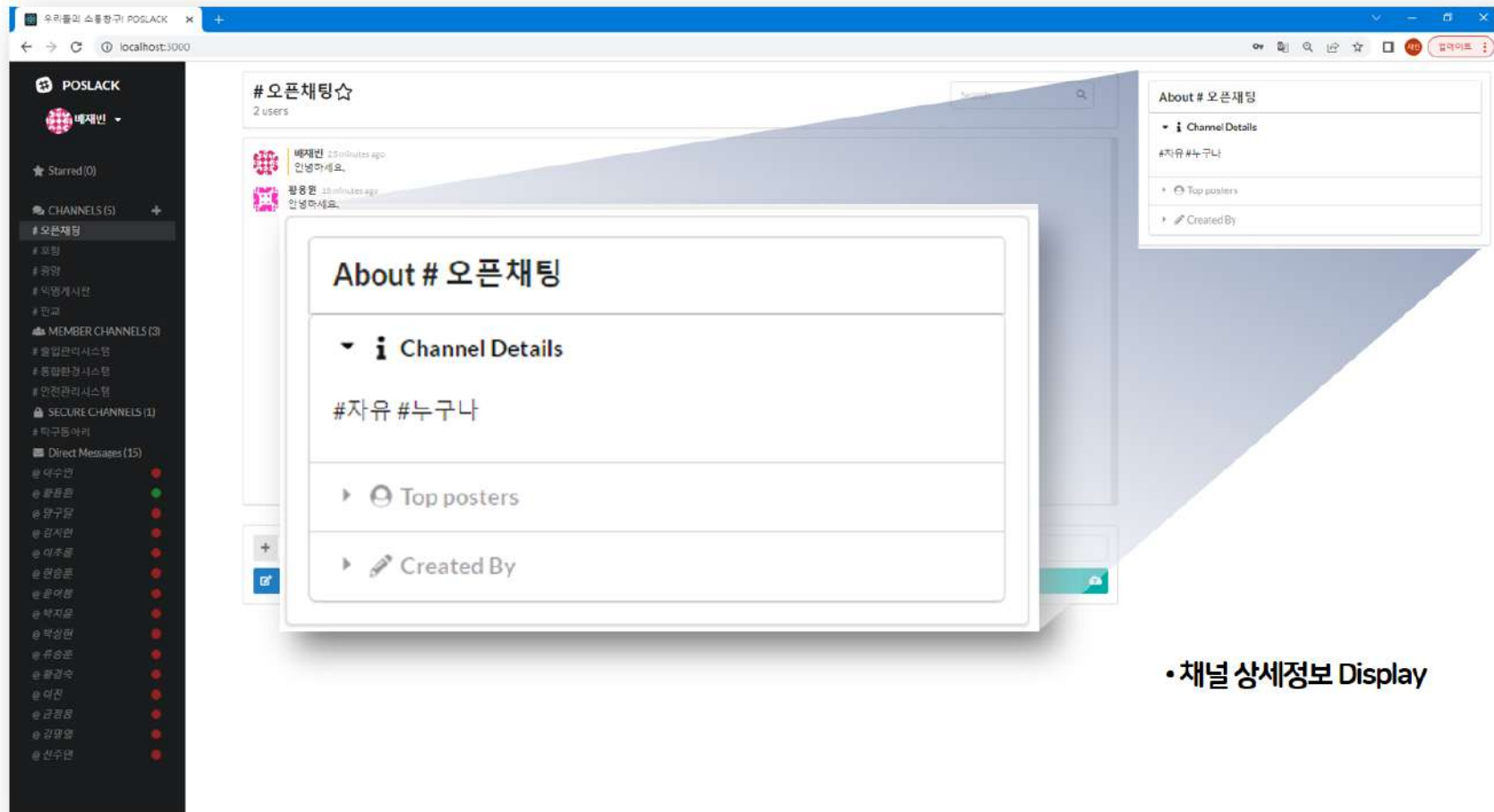
3-3. Side Panel > Direct Messages



3-4. Message



3-5. MetaPanel > Channel Details



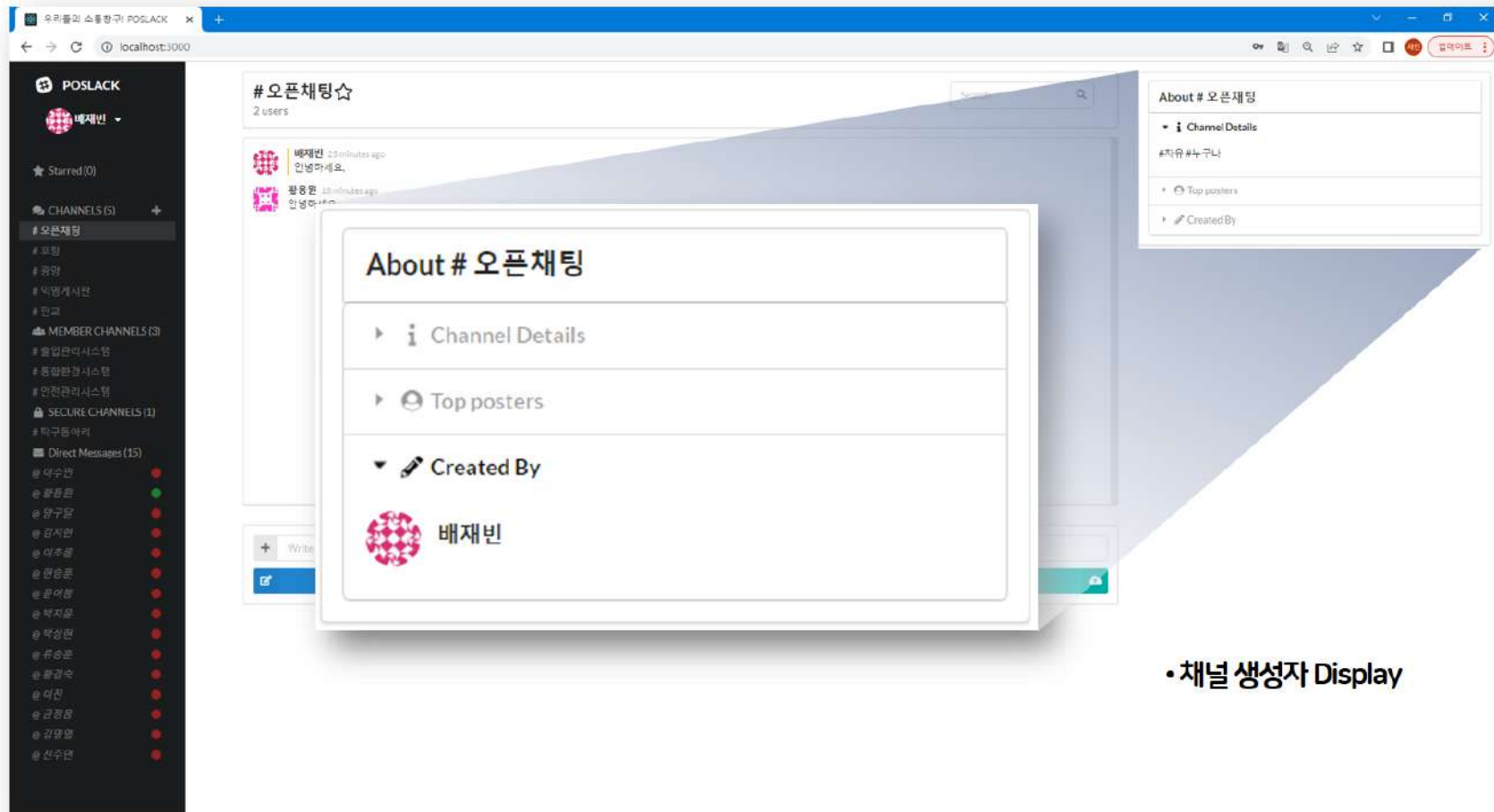
3-5. MetaPanel > Top posters

The screenshot displays the POSLACK web interface. On the left is a dark sidebar with navigation options like 'Starred (0)', 'CHANNELS (5)', and 'MEMBER CHANNELS (3)'. The main area shows a chat window for '# 오픈채팅' with 2 users. A modal window titled 'About # 오픈채팅' is open, showing channel details and a list of 'Top posters'. The list includes:

- 배재빈 (User Avatar) 2 posts
- 황응원 (User Avatar) 1 post

Below the list is a 'Created By' section. In the bottom right corner of the interface, there is a text annotation: '채널 사용자 메시지 수 출력'.

3-5. MetaPanel > Created By



• 채널 생성자 Display

3-5. MetaPanel > Members (Private 채널 전용)

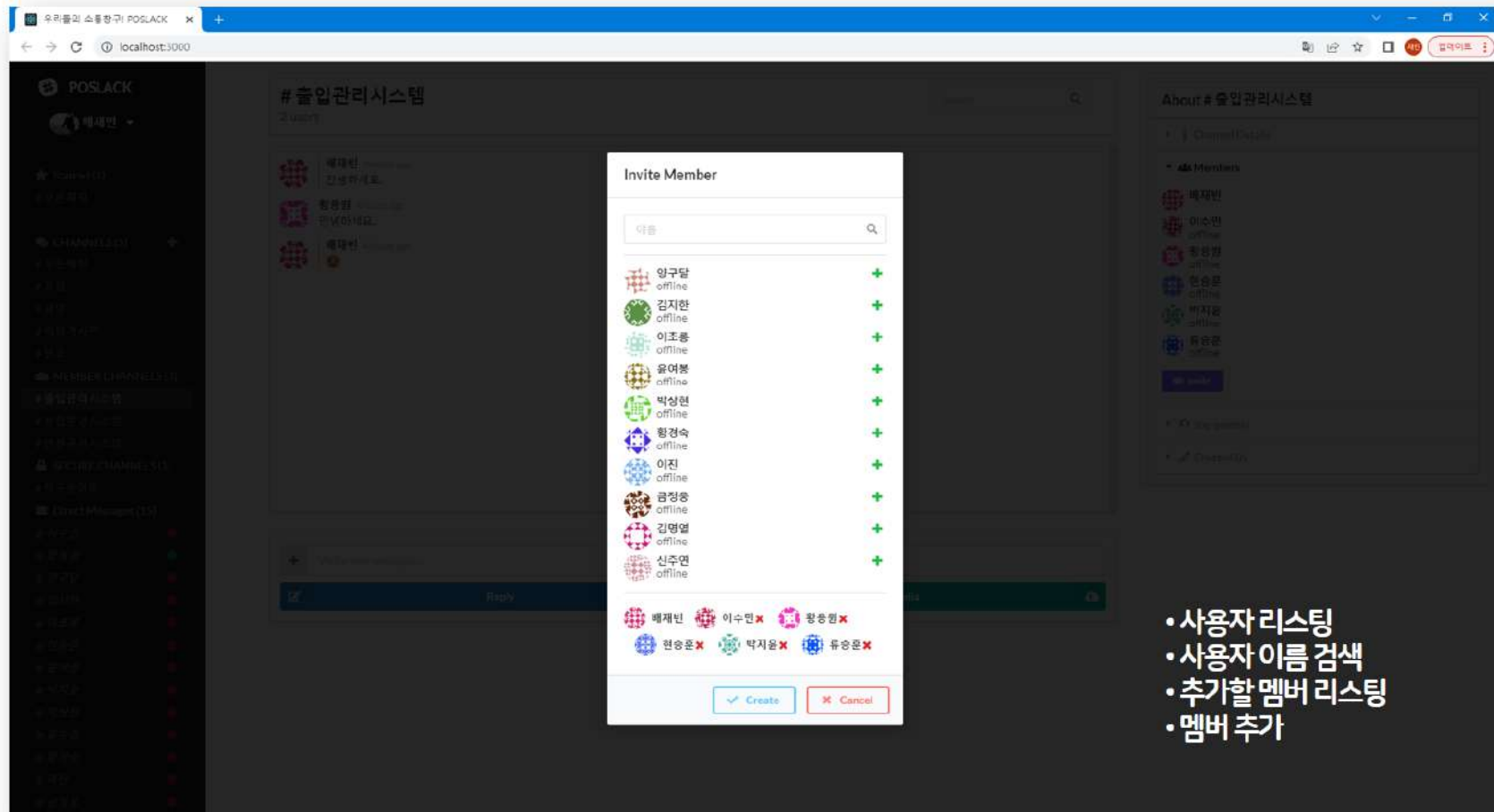
The screenshot displays the POSLACK web interface. A modal window titled 'About #출입관리시스템' is open, showing the following details:

- Channel Details:** #작업 #누구냐
- Members:**
 - 배재빈
 - 이수민 (offline)
 - 황동원 (offline)
 - 현승훈 (offline)
 - 박지윤 (offline)
 - 류승훈 (offline)
- Actions:** Invite, Top posters, Created By

On the right side of the main interface, another 'About #오픈채팅' modal is partially visible, showing channel details and options like 'Top posters' and 'Created By'.

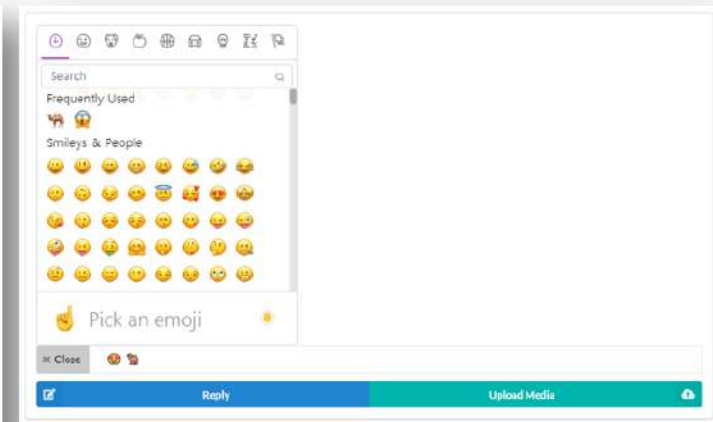
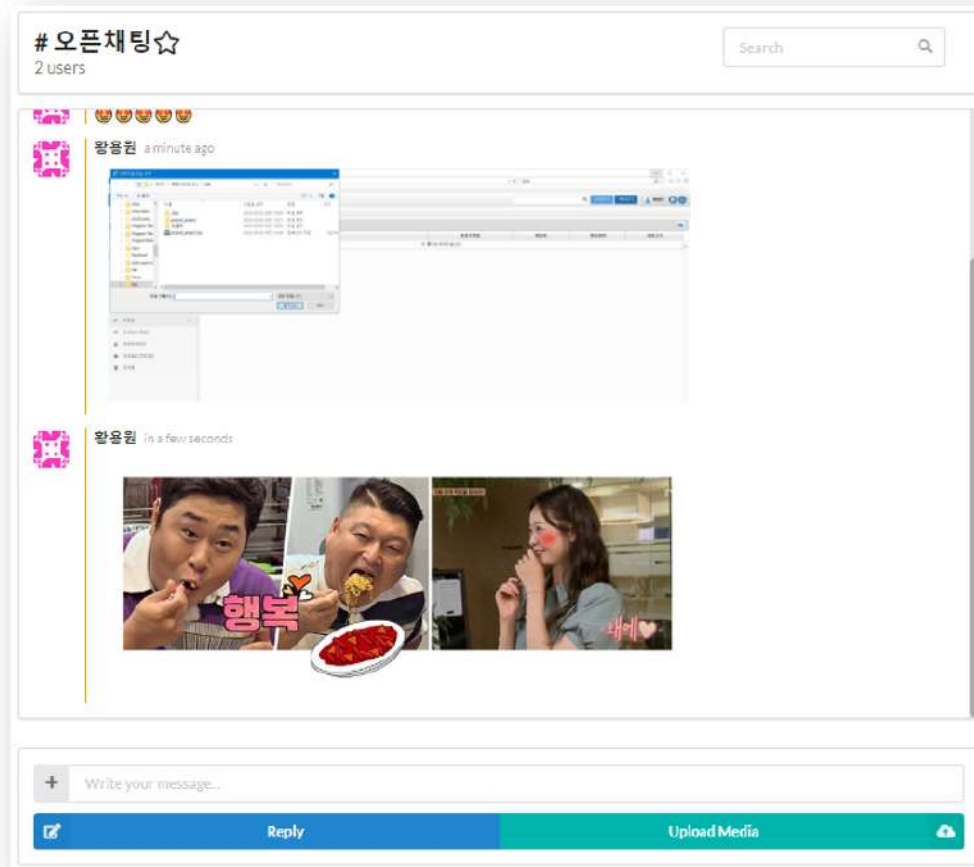
Channel Member Listing
Member Invite
Member Connection Status Listing

3-5. MetaPanel > Members (Private 채널 전용)



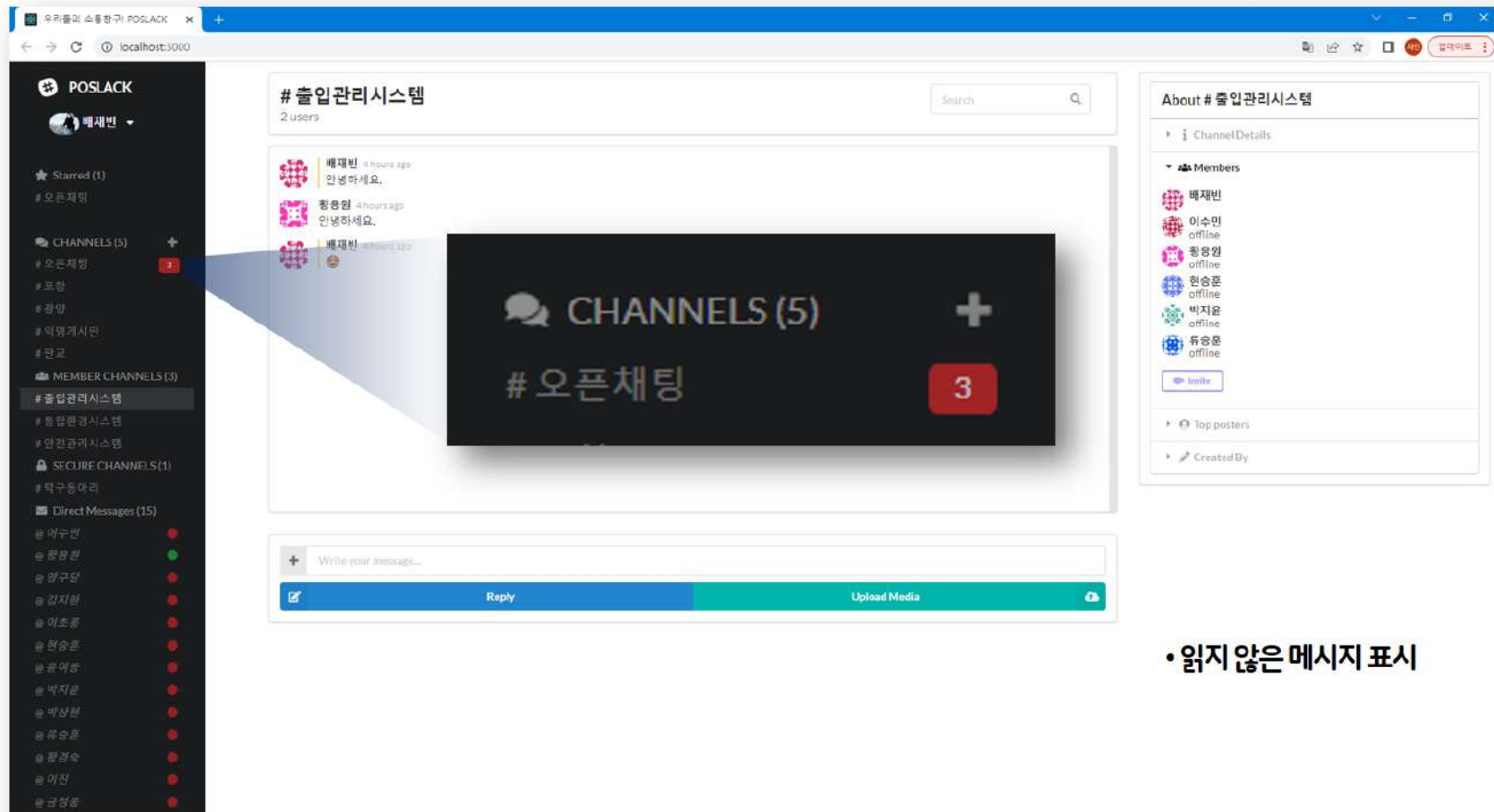
- 사용자 리스팅
- 사용자 이름 검색
- 추가할 멤버 리스팅
- 멤버 추가

3-6. etc



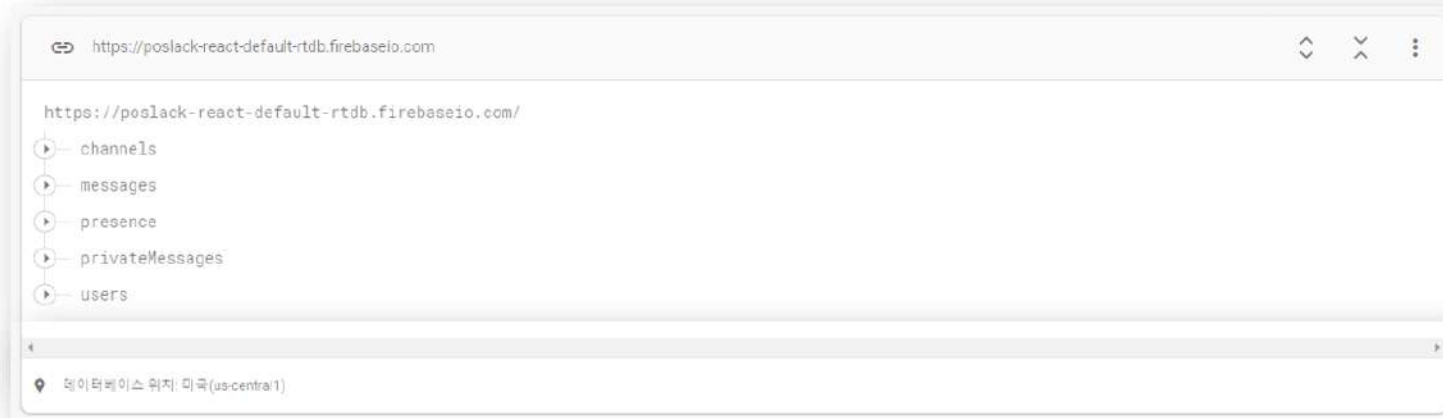
- 이미지 전송
- 이모티콘 전송

3-6. etc

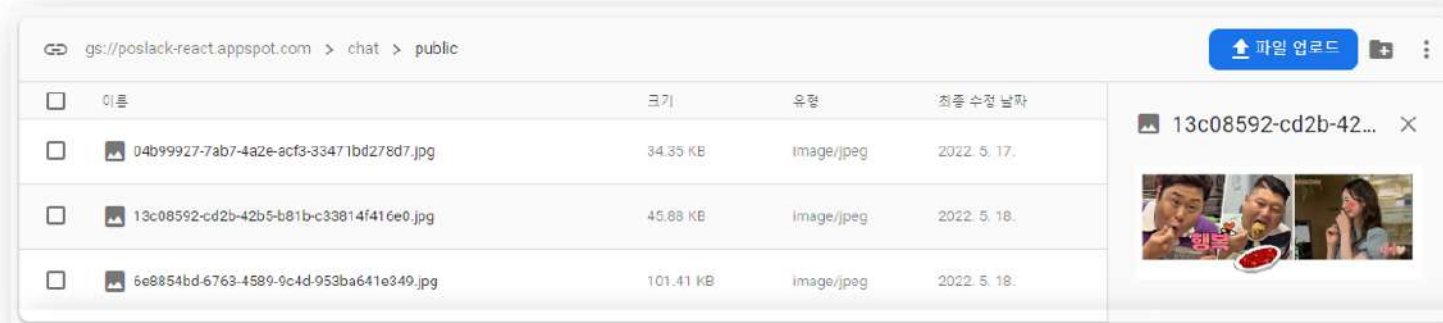


• 읽지 않은 메시지 표시

3-7. Firebase Realtime Database (NoSQL : Data -> Document -> Collection)



3-8. Firebase Storage (File Storage)



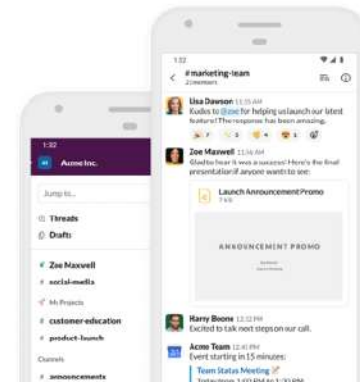
3-9. Firebase Authentication

🔍 이메일 주소, 전화번호 또는 사용자 UID로 검색					사용자 추가	🔄	⋮
식별자	제공업체	생성한 날짜 ↓	로그인한 날짜	사용자 UID			
test16@poscoict.com	📧	2022. 5. 18.	2022. 5. 18.	B4XTM92qcaXXxASHvzTNq00cby...			
test15@poscoict.com	📧	2022. 5. 18.	2022. 5. 18.	5peYeFRcAdhisgVUfHmOGClq9Ef2			
test14@poscoict.com	📧	2022. 5. 18.	2022. 5. 18.	4k4a0grbUwMJCQnvP4qigHqhBe...			
test13@poscoict.com	📧	2022. 5. 18.	2022. 5. 18.	lwEkewh4zSMmhcAven9Q3yhsUJ...			
test12@poscoict.com	📧	2022. 5. 18.	2022. 5. 18.	KrCagJgRmERdaOSOfh22yLuofYV2			
test11@poscoict.com	📧	2022. 5. 18.	2022. 5. 18.	mGxejRG3Sfg7vImWCVzwJIsAawF2			
test10@poscoict.com	📧	2022. 5. 18.	2022. 5. 18.	nqb8ROQTh6WG3PnvI1FB8c082N...			
test09@poscoict.com	📧	2022. 5. 18.	2022. 5. 18.	FF8ta2rvOwUDK16LqvBKATmPOp...			

- 회원가입 폼 Auth에 전달 및 UID 생성
- UID 기반으로 Realtime DB에 User 생성

4. 향후 개선 사항

- React-Native-Navigation을 활용한 Mobile 앱 개발
- 소스 코드 공유 기능 개발
- Firebase Storage(Google Cloud) 속도 개선
- 중복 로그인 제어
- Push Notification
- 메시지를 읽지 않은 사람 수 확인
- 시스템이 확장시 operation 회수 설정



5. 후기

기존에 AWS, RDB(PostgreSQL, OracleDB)로 백엔드를 구현하다가 Firebase를 써보면서 여러가지 차이점을 느꼈다.

환경이 달라지면서 AWS는 비용문제로 사용하지 못하고 빠르게 개발할 수 있는 백엔드를 찾다가 Firebase를 활용하게 되었다.

약간의 오해도 있었다. 대충 읽어보고 AWS-EC2, AWS-S3, AWS-RDS 서비스의 역할을 그대로 할 줄 알았는데 NO-SQL 기반의 Realtime Database 가지고 있었고 Google Cloud 기반의 Storage(NAS 역할) 등 확연한 차이가 있었다.

특히 RDB와 NoSQL의 차이는 많은 혼란이 있었고, 테이블과 스키마의 개념이 없는 대신 유연성이 있었기 때문에 프로토타입 프로젝트에 적합하다는 생각이 되었다.

또한 서버가 아닌 클라이언트가 직접 CUD(set, update, remove) R(on, once)의 역할을 메소드를 호출하면서 수행한다.

일반적으로는 클라이언트가 서버로 DB 조작을 요청하고 서버가 유효성 검사를 통해 처리를 했지만, 기존의 개념과 다르게 클라이언트가 수행한다. (보안관련사항은파이어베이스콘솔에서 직접물을 지정할수있다.)

차이점은 기존에 Spring으로 개발하던 Backend 코드가 필요하지 않고 Front가 직접 서버의 역할도 수행하기 때문에 코드가 길어진다.

검색기능이 기존의 RDB이 어려운 치명적인 단점(NoSQL - Json 구조적 특징)은 있지만 다른 장점도 컸기 때문에 사용할만하다고 생각한다.

진짜 단점. 다중 쿼리 메시지 최근 100개만 가져오는 네비게이팅이 불가능.. 편법써서 해결은 할 수 있을 것 같다. 아니면 cloud function 으로 Update를 특정 document에 append하고 다른 batch job으로 document 호출하는방식?? 로드밸런싱 이슈.

주요 기능 및 구현 방법

주요기능	구현 방법	진행상태
유저 가입 및 로그인	<ul style="list-style-type: none"> • Firebase Authentication => Realtime Database.user 등록 • 로그인할 때마다 사용자 인증 정보가 Firebase Authentication 전송 user.id 토큰(JWT)으로 교환 • 사용자 삭제 / 사용자 비활성화 / 계정에 중대 변화(이메일, 비밀번호) 시 토큰 만료 	완료
유저 리스팅 화면	<ul style="list-style-type: none"> • Realtime Database.user 정보 호출 후 리스트 형식으로 출력 	완료
채팅 화면 & 채팅 메시지 리스팅	<ul style="list-style-type: none"> • 채널 클릭시 key값으로 channels의 정보 중 currentChannel 선정 • currentChannel과 key값이 일치하는 채팅 메시지 리스팅 (해당 채널에 작성된 모든 메시지 출력) 	완료
채팅 메시지 전송	<ul style="list-style-type: none"> • currentUser와 currentChannel의 정보를 기반으로 messages(document)에 채널 key 값과 내용 작성 	완료
채팅방 리스팅	<ul style="list-style-type: none"> • channels(document)를 전체 호출 후 채널 속성에 맞게 Display • Private Channel의 경우 currentUser의 키가 members에 속해있는지 확인 후 Display (filter, .map, .sort, .includes 등 함수 활용) 	완료
채팅방 초대	<ul style="list-style-type: none"> • userList, addMember, memberList 등 state에 user정보 중 필요한 정보를 추출하여 담아서 구현 	완료
접속중인 유저 표시	<ul style="list-style-type: none"> • 접속시 presence(Document) currentUser key와 boolean을 적용 • Logout, 세션 만료 상황에서 .remove 메서드 호출 • True => online(green), False => offline(green) 	완료

주요 기능 및 구현 방법

주요기능	상세내용	진행상태
채팅방 정보	<ul style="list-style-type: none"> • 채팅방을 만든 사람 정보 제공 -> 채널생성시 currentUser 정보 등록 • Top Poster 정보 제공 -> 채널 아이디의 메시지 중 user의 key별로 post횟수 측정 • 채팅방의 상세정보 제공 -> 채널생성시 channelDetail 등록 • 채팅방의 멤버정보 제공 -> Invite 버튼을 통해 채널에 멤버 등록 • 채팅 검색기능 제공 -> 메시지의 채널key와 currentChannel의 key를 조인하여 value를 필터 	완료
채팅방 타입	<ul style="list-style-type: none"> • 4가지 종류의 채팅방 타입 제공 (Direct, Public, Private, Secure) • Direct - channel id가 없기 때문에 개인의 key와 상대의 key를 매핑해서 message 구현 • Public - channel 옵션에서 public 선택 후 생성 • Private - channel 옵션에서 private 선택 후 생성 • Secure - channel 옵션에서 secure 선택 후 비밀번호 설정 및 생성 	완료
즐겨찾기 기능	<ul style="list-style-type: none"> • starred 라는 변수를 통해 user정보에 starred가 true인 채널 key 저장 • 개인마다 starred라는 필드에 채널 key에 따라 채널 출력 	완료
이모티콘 전송 기능	<ul style="list-style-type: none"> • emoji-mart 라이브러리 활용 	완료
읽지않은 메시지 표시	<ul style="list-style-type: none"> • channel.key, user.key, message.key 로 마지막에 읽은 message state에 보관 후 message 이후 생성된 것을 count후 출력 채널 클릭시 message.key 변경되며 notification 사라짐 	완료

주요 기능 및 구현 방법

주요기능	상세내용	진행상태
파일 전송 기능(선택)	<ul style="list-style-type: none">mime-types	완료
프로필 사진 변경 기능	<ul style="list-style-type: none">React-avatar-change 라이브러리 활용	완료