Beddit

Design Document

21812062 김주환

Revision History

Revision date	Version #	Description	Author
2023.05.15	1.0	First Draft	김주환
2023.05.16	1.1	Class diagram	김주환
2023.06.04	1.2	Domain analysis Sequence Diagram State machine diagram	김주환

Contents

1. Introduction

User가 특정 커뮤니티를 이용하는 이유는 단순히 정보수집의 목적도 있지만, '자아 표출의 목적' 도 가지고 있다. 사진이나 글을 업로드하여 다른 User로부터 피드백을 받게 되는데, 이는 인정에 대한 욕구를 채워준다, 그래서 User는 만족감을 얻고 커뮤니티를 이용하게 됩니다.

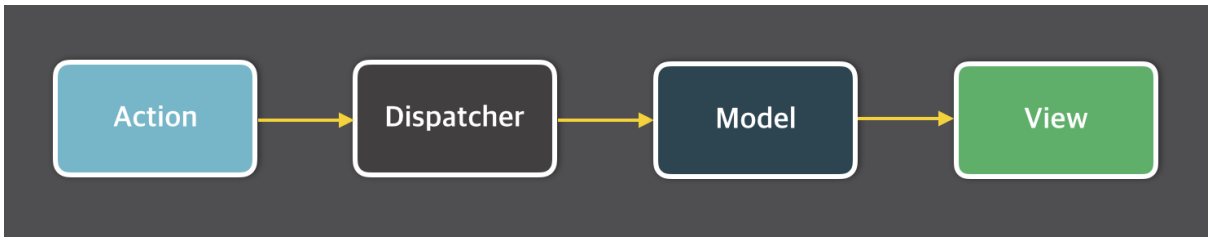
또한 User는 커뮤니티 그룹이 세분화되어 이전과 다르게 자신이 관심있는 공통 주제에 대한 소통을 세분화되어 있는 커뮤니티에서 같은 집단에 속해져 있는 듯한 느낌을 받으며 유대 관계 형성에 도움을 받기도 한다. 기업이 서비스와 함께 커뮤니티를 제공해야 하는 이유는 고객의 체류 시간, 특정 목적이 없어도 해당 앱을 습관적으로 접속한다. SNS를 하듯, 특별한 이유 없이 해당 앱을 자주 방문하고 오래 머물 수 있다.

중고거래 앱 당근마켓은 중고거래 플랫폼에서 지역 공동체 플랫폼으로 발전하고 있다. 지역을 기반으로 중고거래를 할 수 있게 한 이유는 해당 중고거래를 하면서 모인 이용자들을 커뮤니티로 이끌기 위해 서였다. 당근마켓의 동네생활 서비스는 2020년 9월에 오픈한 서비스로, 이웃끼리 유용한 동네정보, 소식 또는 소소한 일상을 나누면서 자유롭게 소통하는 곳으로 즉 하나의 지역기반 커뮤니티로 발전했다.

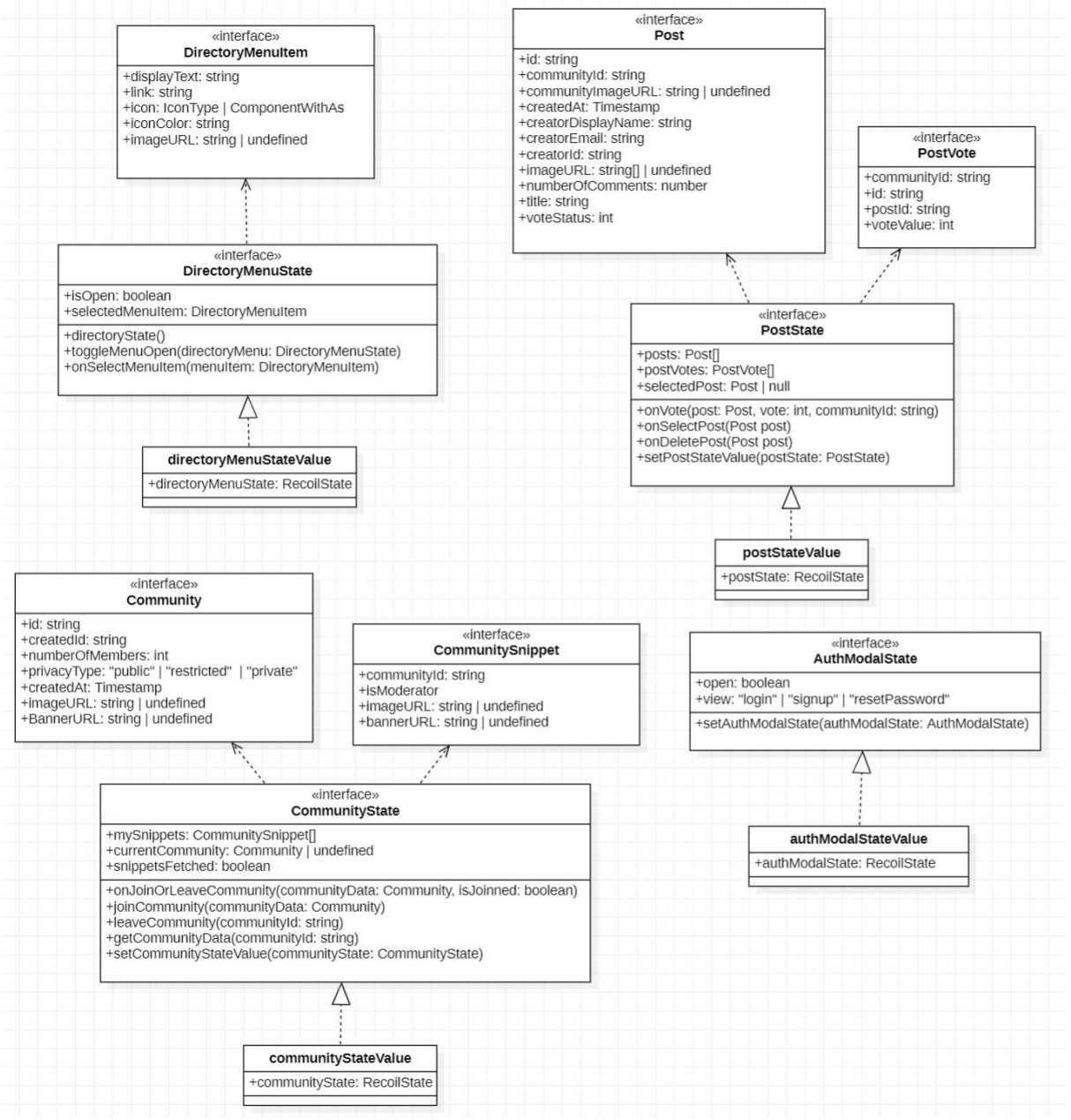
비단 당근마켓 뿐만 아니라 우리 주변에는 다양한 커뮤니티가 존재한다. 대표적으로 주로 대학생이 이용하는 에브리타임, 직장인들이 이용하는 블라인드, 기혼 여성이 주로 이용하는 맘 카페 등이 대표적이다. 이 들을 이용한 커뮤니티 마케팅을 이용하여 잠재 고객 혹은 타겟 고객에 대한 공략이 가능해진다.

2. Class Diagram

본 프로젝트는 MVC 디자인 패턴이 아닌 Flux 디자인 패턴으로 구현 되어있고 사용자의 Action에 따라 Dispatcher가 Store(Model)의 상태를 변경하고 View에 반영하는 방식입니다.



따라서 해당 프로젝트의 Flux 패턴을 Class diagram으로 표현하기 위해서는 Store(Model)의 State 상태를 정의하는 Interface를 Class diagram으로 나타내었습니다. 해당 Class diagram의 Methods에 정의된 것은 파라미터로 전달된 State를 이전에 저장된 State에서 새롭게 업데이트 및 저장할 State로 Dispatch 하는 기능을 담고 있습니다. (해당 Method는 실제로는 Custom hook의 함수로서 인터페이스 외부에 함수의 형태로 따로 구현되어 있습니다.) 로그인, 회원가입은 컴포넌트 함수로 이루어져 있어서 클래스 다이어그램



으로 나타내는데 한계가 있음으로 따로 class diagram에 나타내진 못했습니다.

[그림 2-1] class diagram

1) Community

Attributes
Id : 커뮤니티가 가지는 고유한 값 (Identity) createdId : 커뮤니티를 개설한 유저의 UID 값 numberOfMembers : 커뮤니티에 가입된 멤버 수 privacyType : 커뮤니티의 공개 여부 createdAt : 커뮤니티가 생성된 날짜와 시각 imageURL : 커뮤니티가 가지는 이미지 링크 주소 BannerURL : 커뮤니티가 가지는 배너 링크 주소
Methods

2) CommunitySnippet

Attributes
communityId : 커뮤니티가 가지는 고유한 값 (Identity) isModerator : 커뮤니티를 개설한 유저의 UID 값 imageURL : 커뮤니티가 가지는 이미지 링크 주소 BannerURL : 커뮤니티가 가지는 배너 링크 주소
Methods

3) CommunityState

Attributes
mySnippets : 사용자가 가입한 커뮤니티 목록 상태 CommunitySnippet 배열 currentCommunity : 사용자가 어느 커뮤니티에 접속하고 있는지 상태를 나타내는 Community 객체 snippetsFetched : 사용자가 커뮤니티에 로그인을 하고 커뮤니티 상태가 업데이트 되었는지 여부
Methods
onJoinOrLeaveCommunity(communityData: Community, isJoinned: boolean) : 매개변수로 입력받은 커뮤니티에 가입했는지 여부를 확인후 가입이 안되어 있다면 가입을 하고, 가입이 되어 있다면 탈퇴를 수행한다. joinCommunity(communityData: Community) : 매개변수로 입력받은 커뮤니티에 가입을 수행한다. leaveCommunity(communityData: Community)) : 매개변수로 입력받은 커뮤니티에 탈퇴를 수행한다.

getCommunityData(communityId: string) : 매개변수로 입력받은 커뮤니티 ID를 통해 데이터베이스에서 해당 커뮤니티의 정보를 불러오고 그 정보로 사용자의 currentCommunity 상태를 업데이트한다. setCommunityStateValue(communityState: CommunityState) : 매개변수로 입력받은 커뮤니티 상태를 사용자의 CommunityState로 업데이트 한다.

4) Post

Attributes
Id : 게시글이 가지는 고유한 값 (Identity)
communityId : 게시글이 올려진 커뮤니티의 Id 값
communityImageURL : 게시글이 올라간 커뮤니티의 이미지 URL 주소
createdAt : 게시글이 생성된 날짜
title: 게시글의 제목
body: 게시글의 내용
numberOfComments: 해당 게시글의 작성된 댓글 수
voteStatus: 사용자의 게시글 좋아요(투표) 여부
imageURL: 게시글에 첨부된 이미지 링크 주소
creatorDisplayName : 게시글을 작성한 작성자의 닉네임 또는 이름
creatorEmail : 게시글을 작성한 작성자의 이메일 주소
creatorId : 게시글을 작성한 작성자의 UID
Methods

5) PostVote

Attributes
Id : PostVote가 가지는 고유한 값 (Identity)
communityId : 게시글이 올려진 커뮤니티의 Id 값
postId : 게시글이 가지는 고유한 값 (Identity)
voteValue : 사용자가 해당 게시글에 대한 투표 여부를 나타낸다.
Methods

6) PostState

Attributes
posts : 게시글의 상태를 저장하기위한 Post 배열
postVotes : 게시글의 투표 상태를 저장하기위한 PostVote 배열 속성

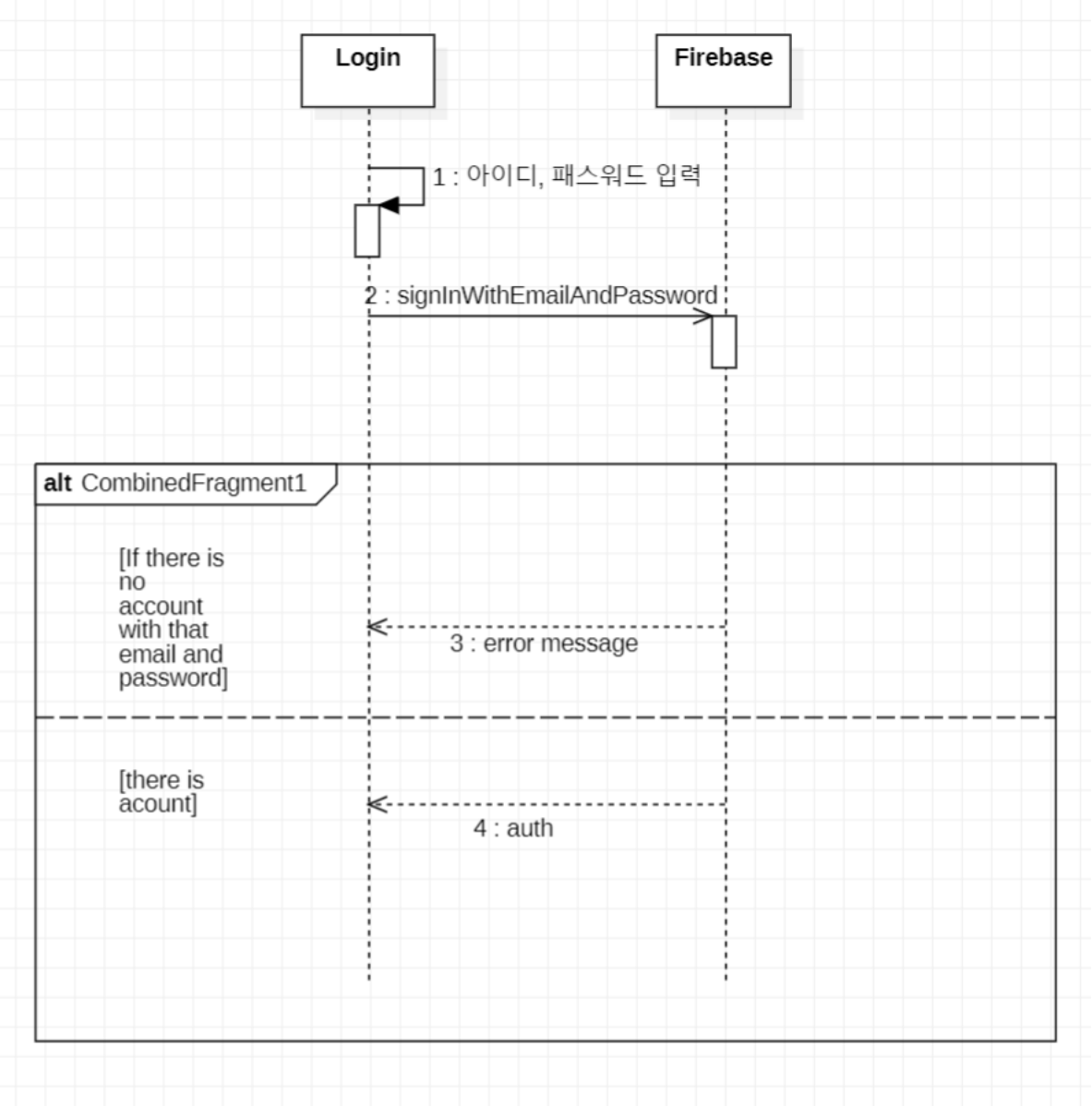
selectedPost : 현재 선택된 게시글이 있는지 여부를 나타내는 상태 속성
Methods
setPostStateValue(postState: PostState) : 매개변수로 입력받은 게시글 상태를 사용자의 PostState로 업데이트 한다. onVote(post: Post) : 게시글에 투표 기능을 제공 onSelectPost(post: Post) : 게시글의 상세 페이지 보기 onDeletePost(post: Post) : 게시글 삭제 기능을 제공

7) AuthModalState

Attributes
open : 사용자의 auth Modal 창이 열려 있는지에 대한 여부를 나타내는 상태 속성입니다. view : 사용자가 어떤 종류의 Modal 창을 선택하였는지에 대한 여부를 나타내는 상태 속성입니다.
Methods
setAuthModalState(authState: AuthState) : 매개변수로 입력받은 AuthModal 상태를 사용자의 AuthModal 로 업데이트 한다.

3. Sequence Diagram

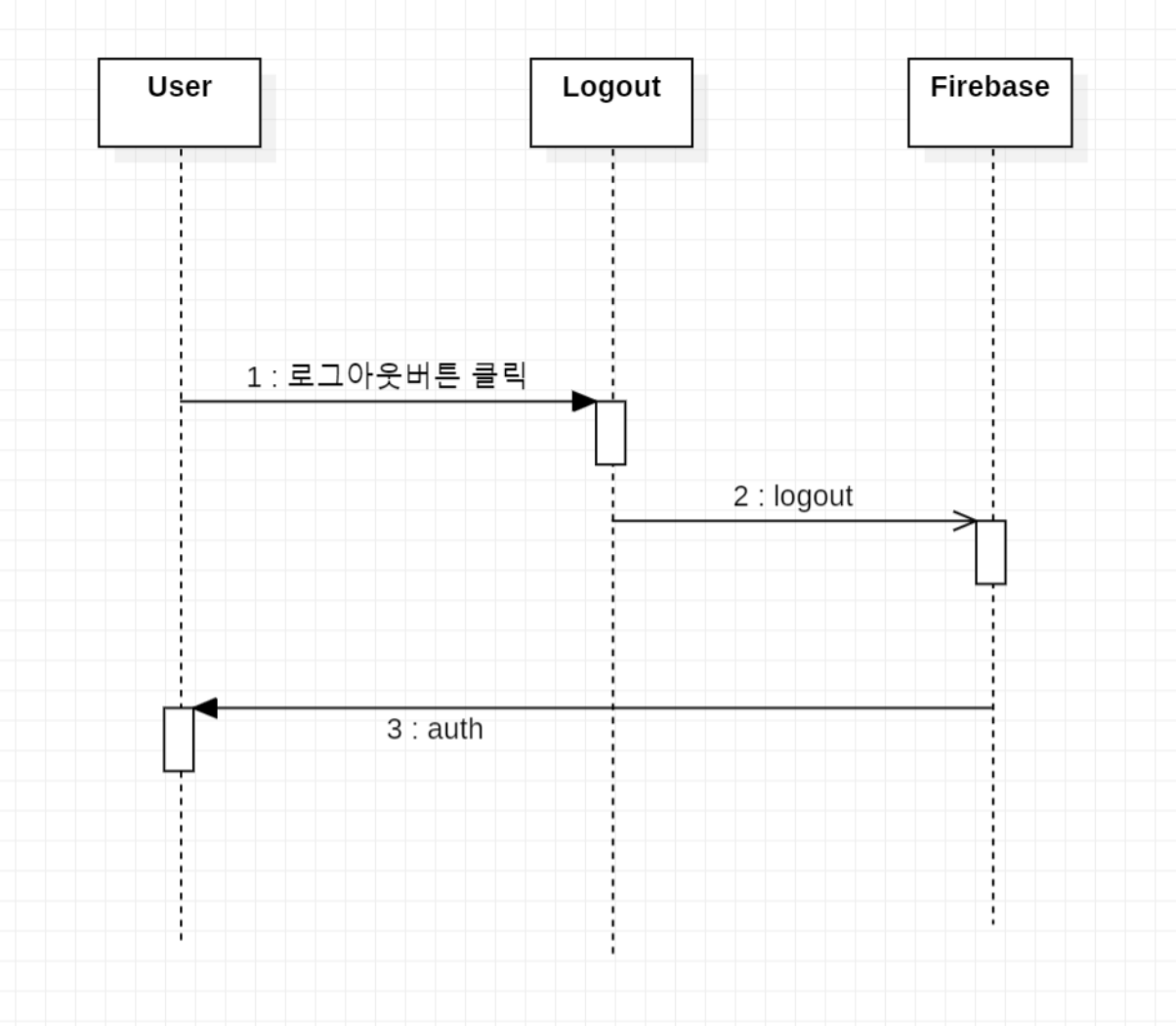
(다음으로 정의된 인터페이스는 클라이언트의 action 및 Firebase DB 서버에서 가져온 데이터를 이용하여 상태를 dispatch 하여 클라이언트에게 보여줍니다.)



1) Login

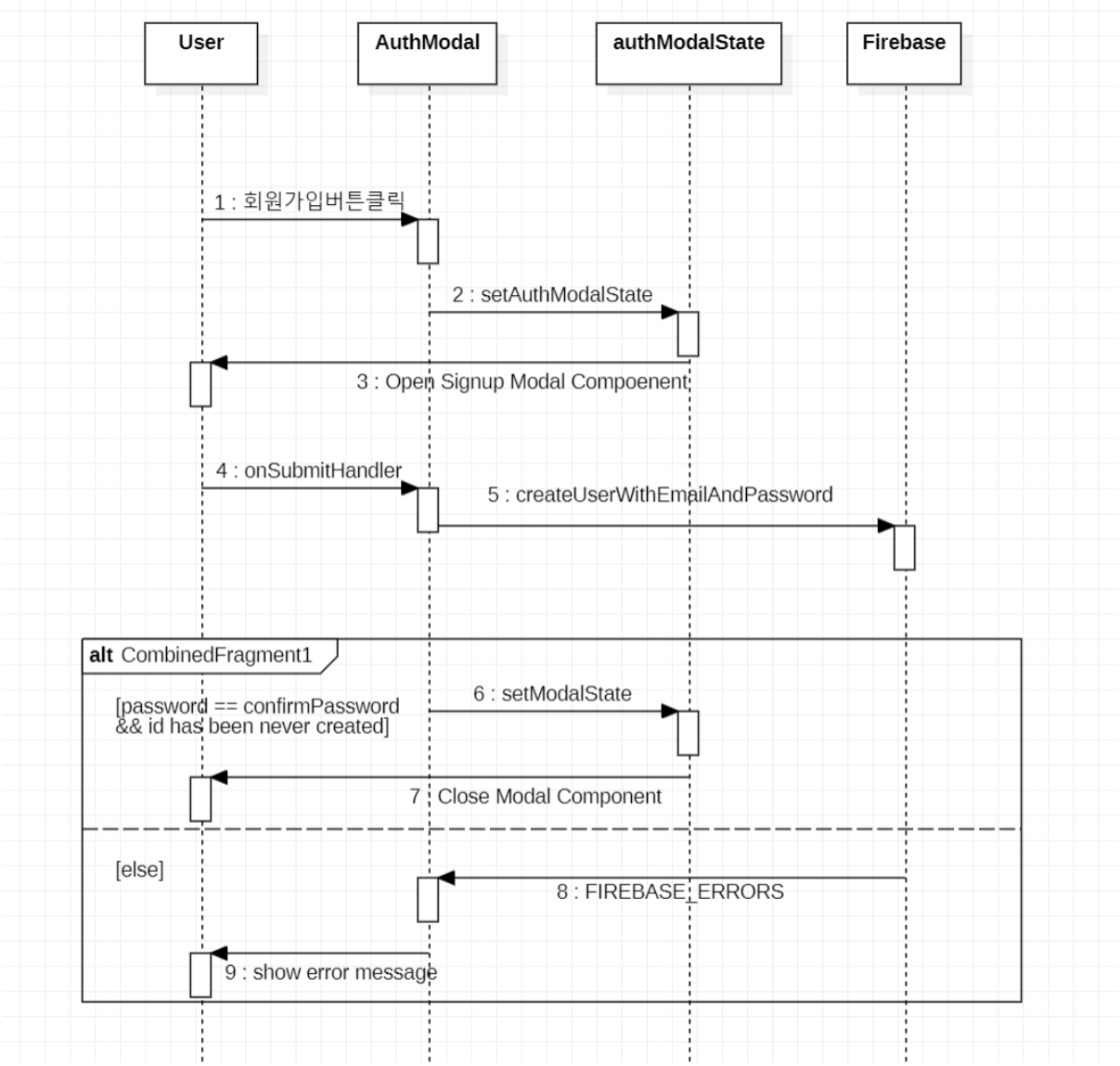
유저가 아이디와 패스워드를 입력하고 로그인 버튼을 클릭합니다.

Login 컴포넌트에서 signInWithEmailAndPassword 함수를 실행하고 Firebase에서 해당 아이디와 비밀번호가 일치한다면 auth를 클라이언트에게 반환합니다.



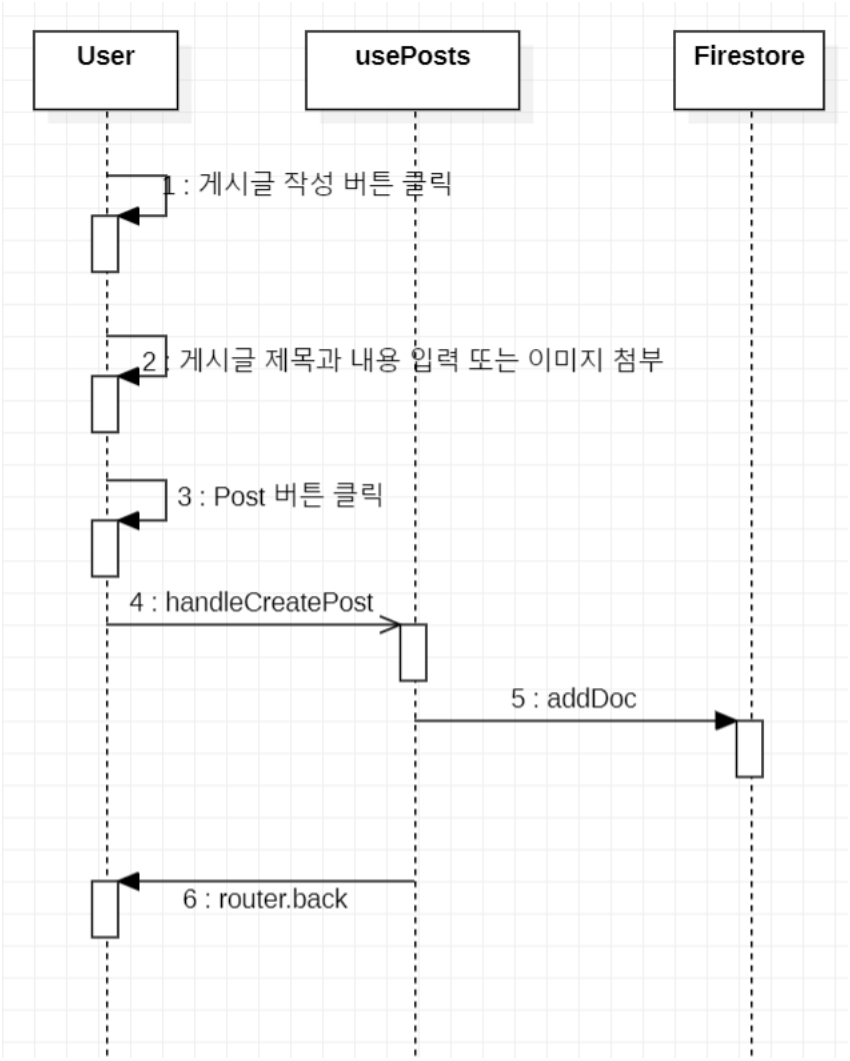
2) Logout

유저가 로그아웃 버튼을 클릭한 다음에 Logout 컴포넌트가 logout 함수를 실행합니다. Firebase에서 현재 로그인 중인 auth를 반환합니다.



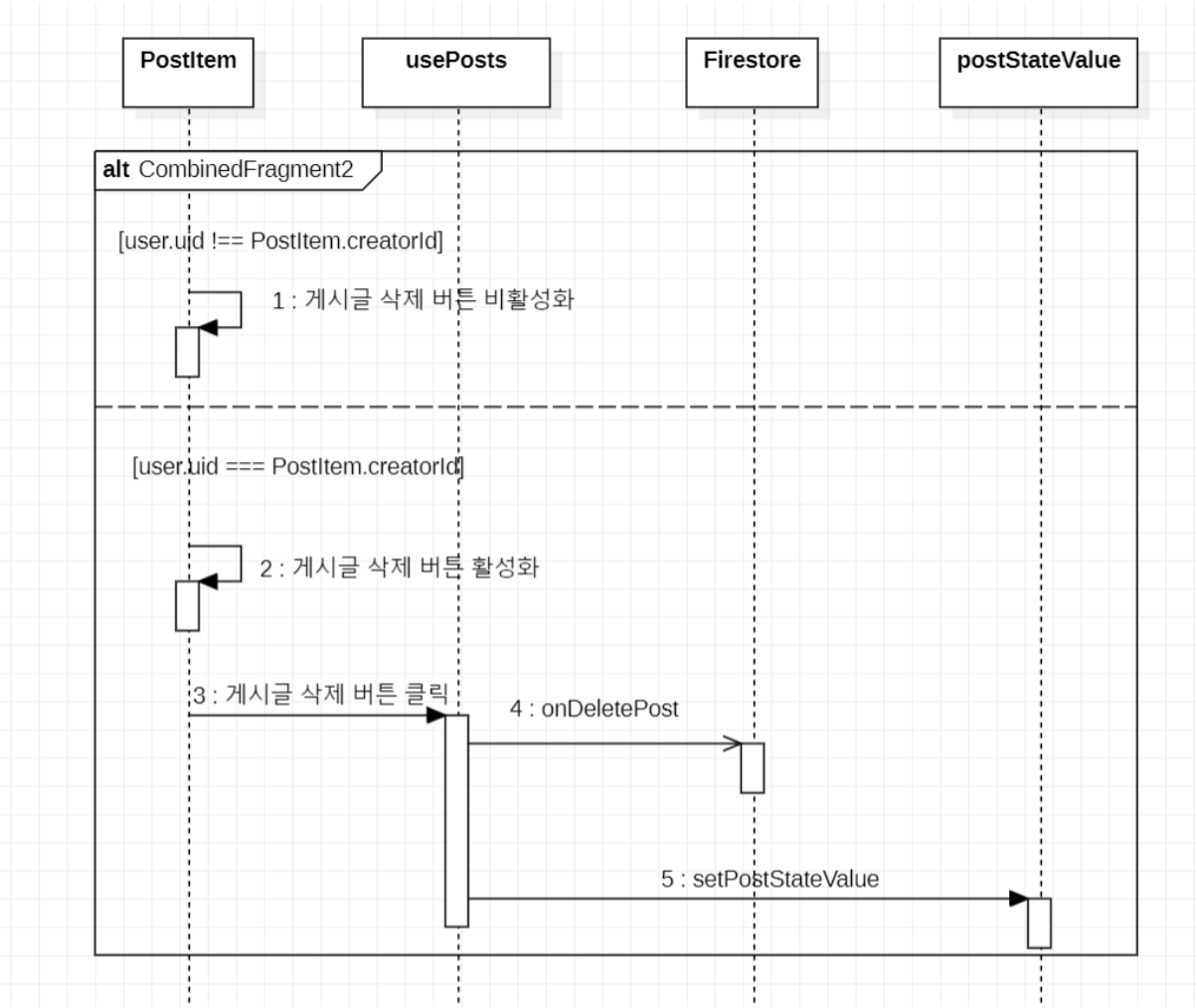
3) Signup

유저가 회원가입 버튼을 클릭한 뒤에 AuthModal 컴포넌트가 authModalState의 상태 속성 중에 view를 “login”으로 상태를 변경하고 유저가 회원가입 Modal에서 아이디와 비밀번호, 비밀번호 확인 입력을 하고 제출을 하게 되면 AuthModal은 Firebase로 유저가 입력한 입력 값을 전달하게 된다. 만약 해당하는 아이디가 존재하거나 비밀번호를 다르게 입력했을 경우 Firebase로부터 오류 메세지를 전달받게 되고, 그렇지 않다면 정상적으로 회원가입이 성공하고 AuthModal 컴포넌트는 authModalState의 상태 속성 중에 open 값을 false로 하여 해당 창을 닫는다.



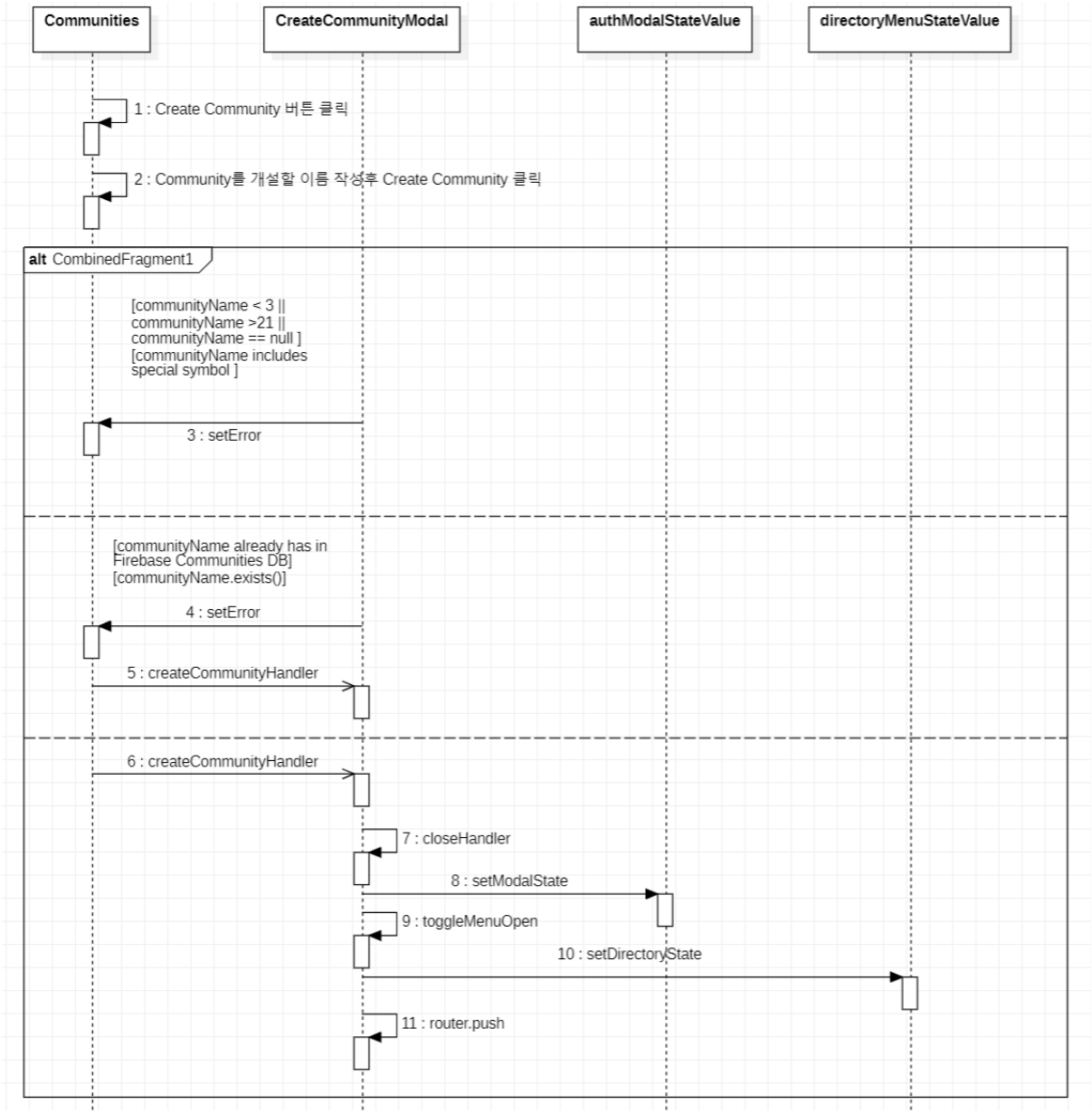
4) CreatePost

유저가 게시글 작성 버튼을 클릭하고 게시글 제목과 내용 또는 이미지를 첨부하고 Post 버튼을 클릭하여 게시글을 제출하게 되면 usePost 컴포넌트에서 handleCreatePost 비동기 함수를 통해 Firestore에 해당 게시글의 제목과 내용을 데이터를 저장하게 되고 usePost컴포넌트는 유저를 router.back 함수를 통해 이전화면으로 되돌아가도록 한다.



5) DeletePost

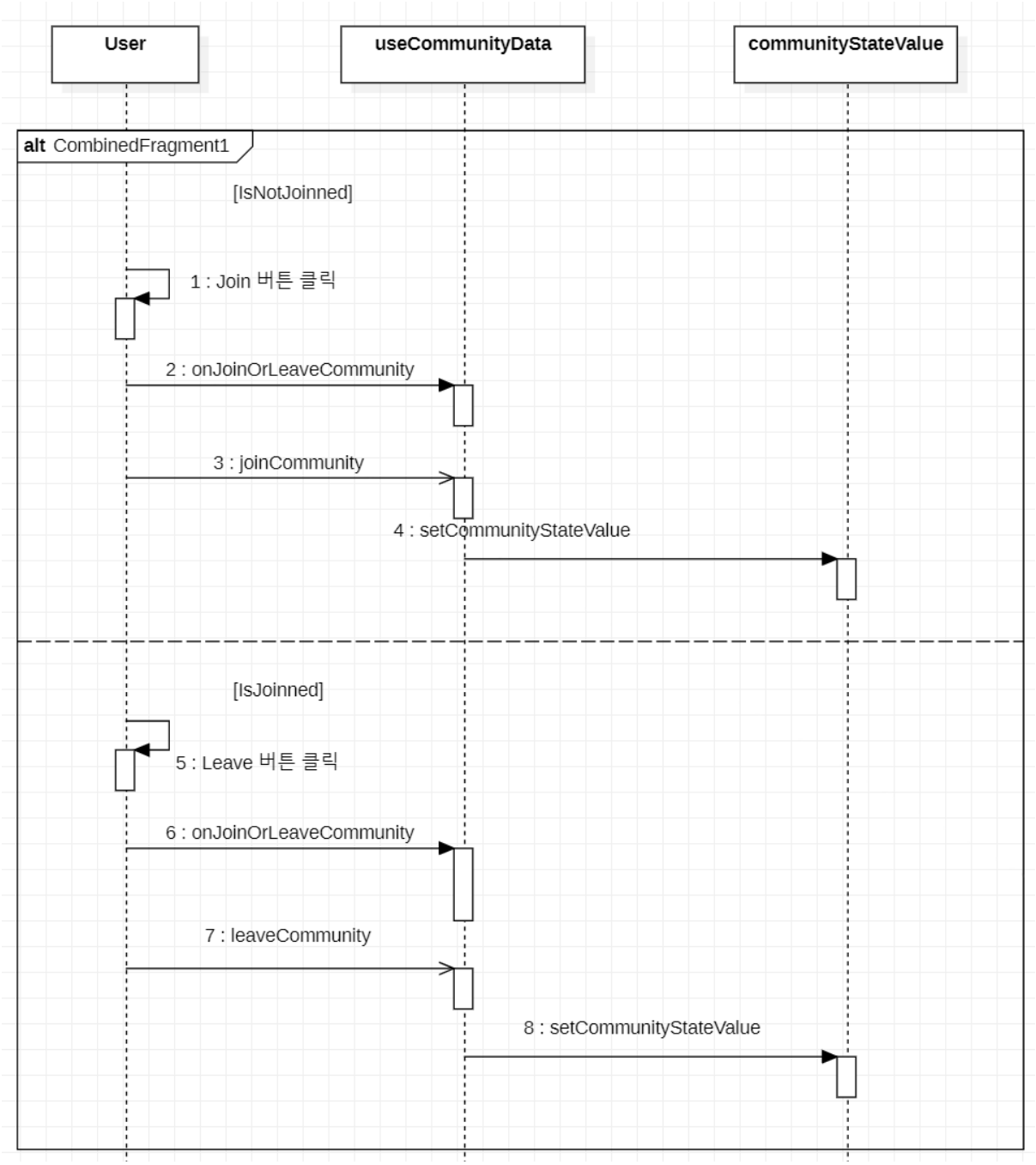
유저의 uid가 postStateValue의 posts의 creatorId 와 같다면 삭제버튼이 활성화가 되며 유저가 삭제 버튼을 클릭 시에 usePost 컴포넌트에서 onDeletePost 비동기 함수가 실행되고 해당 함수는 Firestore에서 해당되는 게시글의 id와 같은 게시글을 삭제합니다.



6) Create Community

사용자가 Create Community 버튼을 클릭합니다. 해당 관련 Modal이 나타나고 사용자가 해당 Modal에 개설하고 자하는 Community 이름을 작성 후 Create Community 버튼을 클릭합니다. 작성한 커뮤니티 이름이 Firebase Communities에 존재한다면 오류를 반환하고 커뮤니티 이름이 3자미만이거나, 21자를 초과 및 이름을 기입하지 않았다면 CreateCommunityModal 컴포넌트 로부터 오류 메시지를 받습니다.

어떠한 조건에도 해당되지 않는다면, 해당하는 이름의 커뮤니티가 createCommunityHandler 함수에 의해 정상적으로 만들어지고 Modal 창이 closeHandler 함수에 의해 ModalState 상태 값이 바뀌면서 닫히게 되고, toggleMenuOpen 함수로 인해 현재 접속중인 커뮤니티 상태가 변경됩니다 다음으로 router.push로 인해 해당하는 커뮤니티로 이동하게 됩니다.

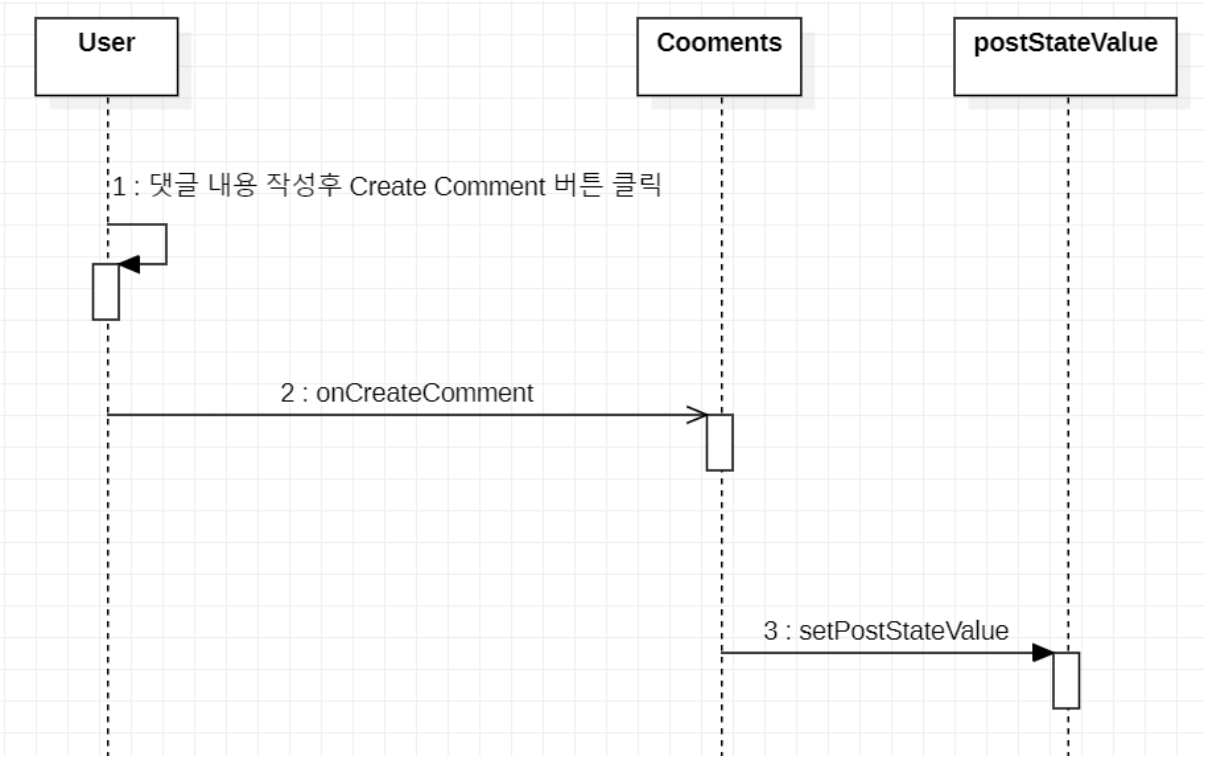


8) Join Community or Leave Community

유저가 해당 커뮤니티에 이미 가입된 경우라면 Leave 버튼이 표시되고 해당 버튼을 클릭하게 됩니다. 이 경우에는 useCommunityData 컴포넌트, onJoinOrLeaveCommunity 함수에서 leaveCommunity 함수를 실행하여 Firebase에 해당 유저의 uid를 communities DB에서 삭제하게 됩니다.

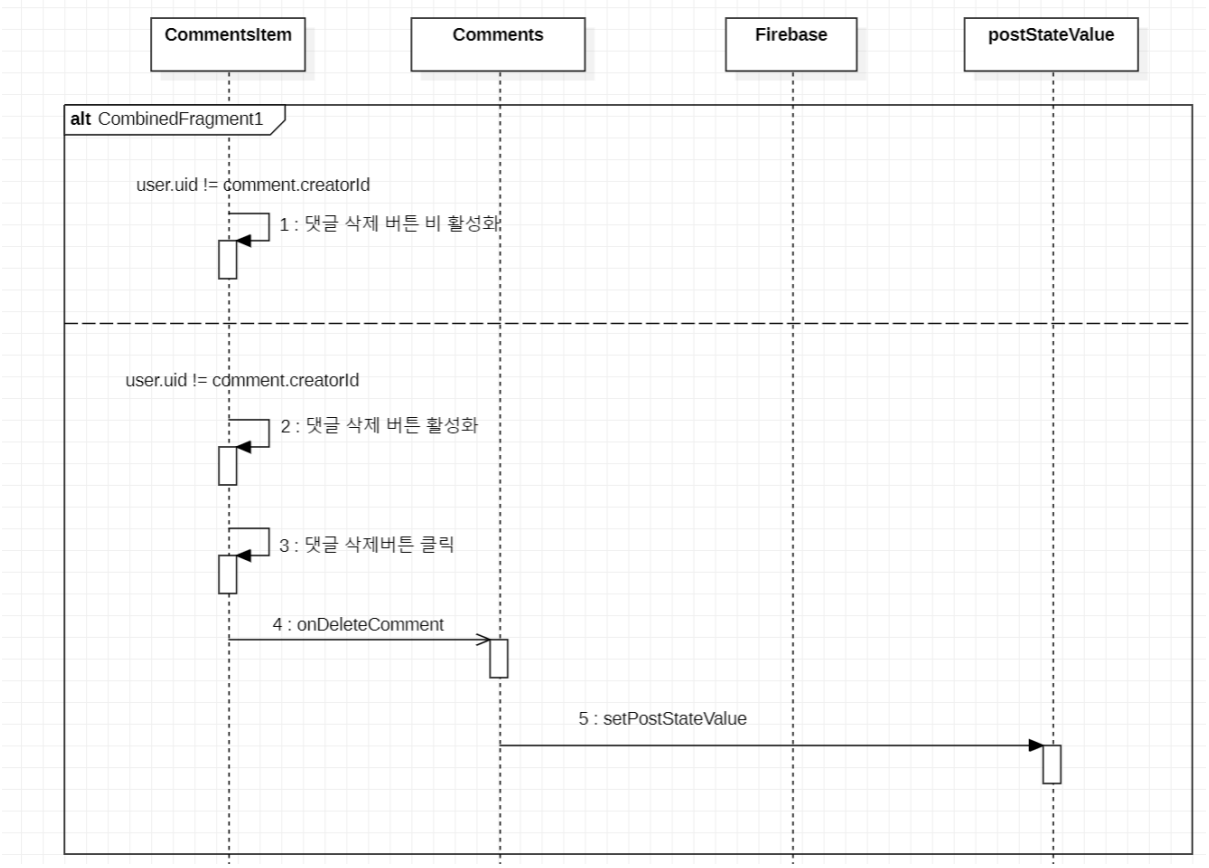
가입이 안된 경우라면 Join 버튼이 표시되고 해당 버튼을 클릭하게 됩니다. 이 경우에는 useCommunityData 컴포넌트에서 onJoinOrLeaveCommunity 함수에서 joinCommunity 함수를 실행하여 Firebase에 해당 유저의 uid를 communities DB에 기입하게 됩니다.

이 두 과정이 끝난 다음에는 useCommunityData 컴포넌트에서 setCommunityStateValue 함수를 통해 communityStateValue에 communitySnippet의 상태를 변경합니다 communitySnippet의 상태를 통해 유저가 해당 커뮤니티에 가입했는지 여부를 나타냅니다.



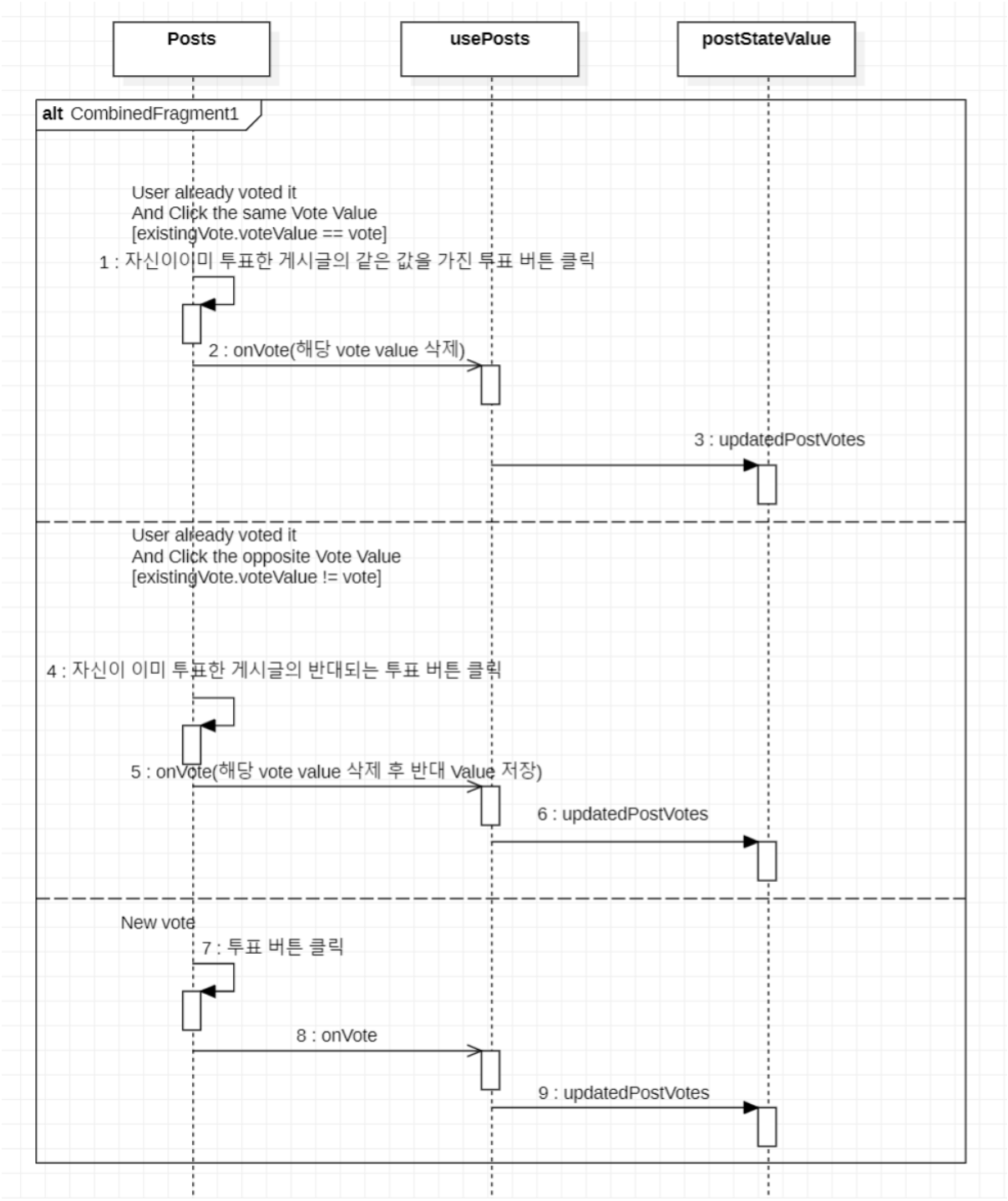
9) Create Comment

유저가 댓글내용을 작성 후 Create Comment 버튼을 클릭하게 되면 Comments 컴포넌트에서 onCreateComment 함수를 동작하여 Firebase에 해당 댓글의 내용과 유저의 uid를 Post DB에 저장한다. 그 후 Comments 컴포넌트는 setPostStateValue를 통해 postStateValue의 selectedPost의 numberOfComments 상태를 조정한다. 해당 상태 값을 통해 게시글의 댓글 개수를 나타낸다.



10) Delete Comment

유저가 작성한 댓글이라면 댓글 삭제 버튼이 활성화가 되어 클릭 할 수 있고 그렇지 않다면 비활성화 된다. 삭제 버튼을 클릭한다면 Comments 컴포넌트는 setPostStateValue를 통해 postStateValue의 selectedPost의 numberOfComments 상태를 조정한다. 해당 상태 값을 통해 게시글의 댓글 개수를 나타낸다.



11) Voting

사용자가 만약 이미 투표한 게시글이고, 해당 voteValue가 이미 사용자가 투표한 Value와 같다면 onVote 함수에서 해당 vote Value를 삭제하고 Firebase DB에 반영한다. 사용자는 해당 투표를 취소한 것과 같다.

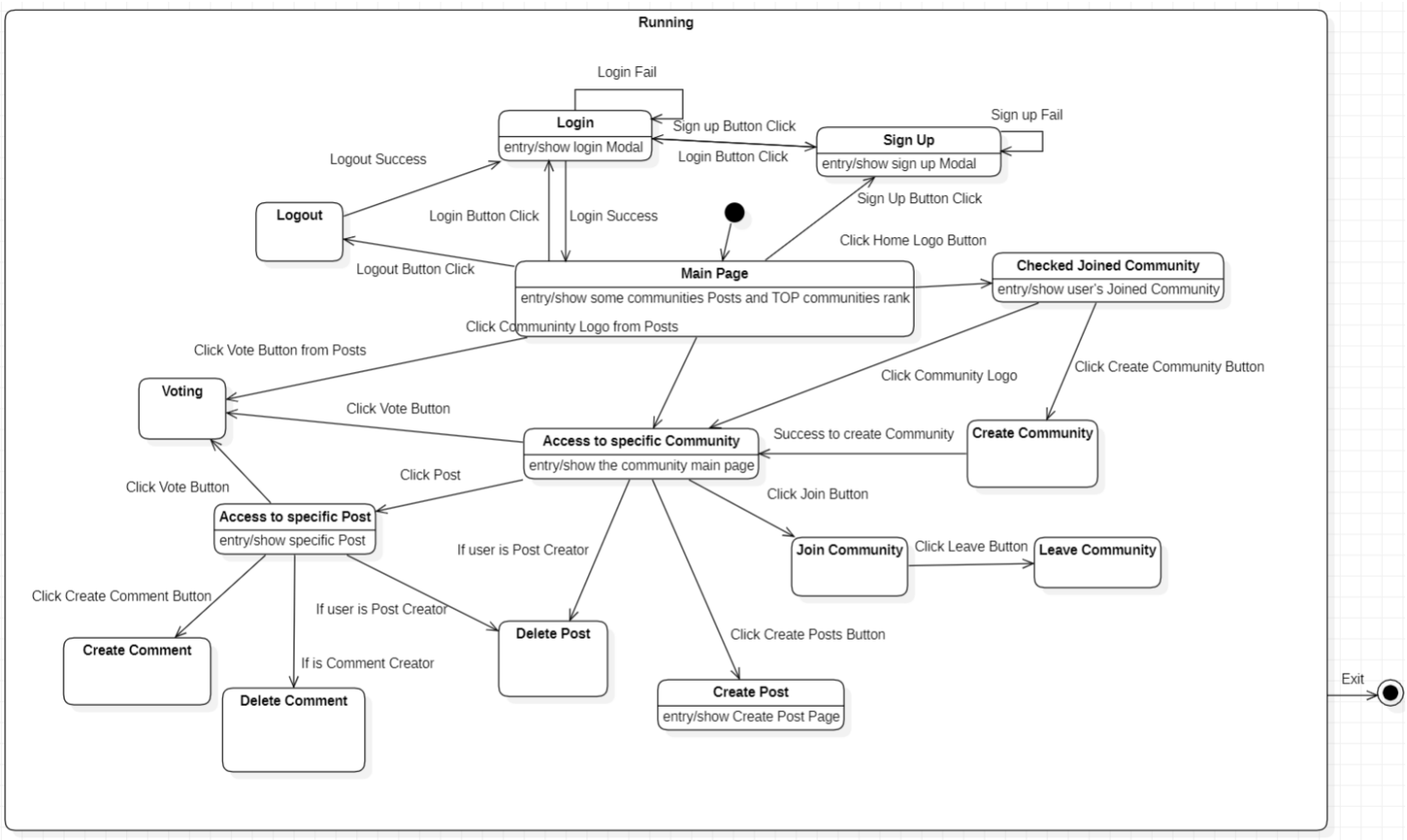
다음은 사용자가 만약 이미 투표한 게시글이고, 해당 voteValue가 이미 사용자가 투표한 Value와 다르

다면 onVote 함수에서 해당 vote Value를 삭제하고 새로운 voteValue를 반영하고 이를 Firebase DB에 반영한다. 사용자는 이전에 투표를 취소하고 새로운 vote Value에 투표한 것과 같다.

다음은 사용자가 만약 이전에 투표한적 없는 게시글이라면 사용자가 클릭한 vote Value에 따라 onVote 함수가 Firebase DB에 반영한다.

해당 과정이 끝난 뒤에 onVote 함수는 공통적으로 updatedPostValues 함수를 통해 현재의 postStateValue 상태를 조정한다.

4. State machine diagram



먼저 사용자가 시스템 실행 시 Beddit의 Main Page에 처음으로 접속하게 되고, 로그인 버튼을 눌러 로그인을 시도하거나 메인 화면에서 Register버튼을 눌러 바로 회원가입을 시도할 수 있다. 로그인이 성공하면 메인 화면으로 되돌아오고, 로그인을 안 한 비 회원인 유저는 커뮤니티를 보는 건 가능하지만 커뮤니티를 생성하거나, 가입하거나, 게시글 및 댓글을 작성, 삭제를 하지 못한다. 반면, 회원인 경우에 메인 페

이지에서 자신이 가입한 커뮤니티 목록을 확인하여 새로 커뮤니티를 생성하거나, 자신이 가입한 커뮤니티 중에 접속 하고싶은 커뮤니티를 클릭하여 특정 커뮤니티로 접속이 가능하다. 또는 Main Page에 올라온 게시글의 커뮤니티 로고를 클릭하여 해당 게시글이 작성된 특정 커뮤니티로 접속이 가능하다. 특정 커뮤니티에서는 게시글을 생성하거나, 삭제, 커뮤니티 가입, 탈퇴, 투표 등이 가능하며 업로드 되어진 게시글 중에 하나를 클릭하면 해당 게시글에 대한 상세정보로 접속하게 된다. 상세 정보에서는 해당 게시글에 대한 댓글을 작성 및 삭제할 수 있고 해당 게시글의 투표도 할 수 있다.

5. Implemtation requirements

1) 하드웨어 요구사항

1. CPU : Intel Pentium 4 이상
2. RAM : 1GB 이상
3. HDD / SSD : 1GB 이상

2) 소프트웨어 요구사항

1. OS : Windows 7 이상

6. Glossary

Terms	Description
Flux Design Pattern	Flux는 사용자 입력을 기반으로 Action을 만들고 Action을 Dispatcher에 전달하여 Store(Model)의 데이터를 변경한 뒤 View에 반영하는 단방향의 흐름으로 애플리케이션을 만드는 아키텍처입니다.
Custom Hook	특정 상태관리나 라이프사이클 로직들을 추상화하여 묶어서 재사용이 가능하도록 제작이 가능한 함수를 뜻한다.
Class Diagram	시스템의 클래스, 클래스의 속성, 동작 방식, 객체 간 관계를 표시하여 시스템의 구조를 보여주는 정적 구조 다이어그램입니다.
Sequence Diagram	시퀀스 다이어그램(Sequence Diagram)은 어떠한 순서로 어떤 객체들과 어떻게 상호작용했는지를 표현하는 다이어그램
State (상태)	상태의 정의는 웹 어플리케이션을 render하는데 있어 영향을 미칠 수 있

	는 값이라고 말한다. react에서 페이지가 리랜더링 되는 가장 큰 예시 두 가지는 props와 state 이 있다.

7. References

[React의 상태 정의]

https://velog.io/@ykh0316/ReactState%EC%83%81%ED%83%9C-%EB%9E%80

[클래스 다이어그램의 정의]

https://parkadd.tistory.com/123

[순차 다이어그램의 정의]

https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=myeongdms55&logNo=220971093680

[FLUX 디자인 패턴의 정의]

https://velog.io/@andy0011/Flux-%ED%8C%A8%ED%84%B4%EC%9D%B4%EB%9E%80