Walking Mate

project.D

2016314670 문정연
2014314143 김규용
2018311304 성보경
2016311561 송유호

Contents

Front-End

Implementation - 하단 네비게이션 뷰 및 화면 전환

switch

case walk:

case trip:

case chat:

해당 fragment만 비추고 나머지는 숨기도록 리액션

기존에 띄워진 fragment가 없을 때만 새롭게 띄운다. (초기화 방지)

case trace:

바로 도보 추적 activity 실행

case feed:

해당 activity 자체에 존재하므로 mainLayout을 띄움

Front-End

Implementation - 하단 네이게이션 뷰 및 화면 전환

WalkFragment

앱의 메인화면이므로 앱 시작과 동시에 실행한다.

ChatFragment

사용자가 최초에 chat을 선택했을 때도 로딩을 기다릴 필요가 없도록 앱 시작과 동시에 실행한다.

Implementation - 유저 데이터

```
public class UserData {
    String userid;
    String profileImagebig; //ex) 마이 페이지
    String profileImagesmall; // ex) 채팅
    String appname;

    String nickname;
    String name;
    String age;
    String gender;
    String birthyear;

    String title; //칭호
    Long reliability; //신뢰도
}
```

파이어스토어에 기록되는 필드 그대로
객체를 만들어 로컬에 저장

Implementation - 유저 데이터

getHashmap()

파이어스토어에 전송하기 위한 Hash값을 구성한다.

saveData() / loadData()

유저 정보를 로컬에 저장한다.(신규 회원, 정보 갱신) / 로컬에 저장되어 있는 정보를 불러온다.

encode() / decode()

UserData를 String 형태로 encode한다. String 형태의 UserData를 decode한다.

scanUserData()

저장되어 있는 유저 데이터를 확인하여 자동로그인을 할지 말지 결정한다.

Implementation - 유저 데이터

saveBitmapToJpeg()

유저 프로필 이미지를 로컬에 저장한다.

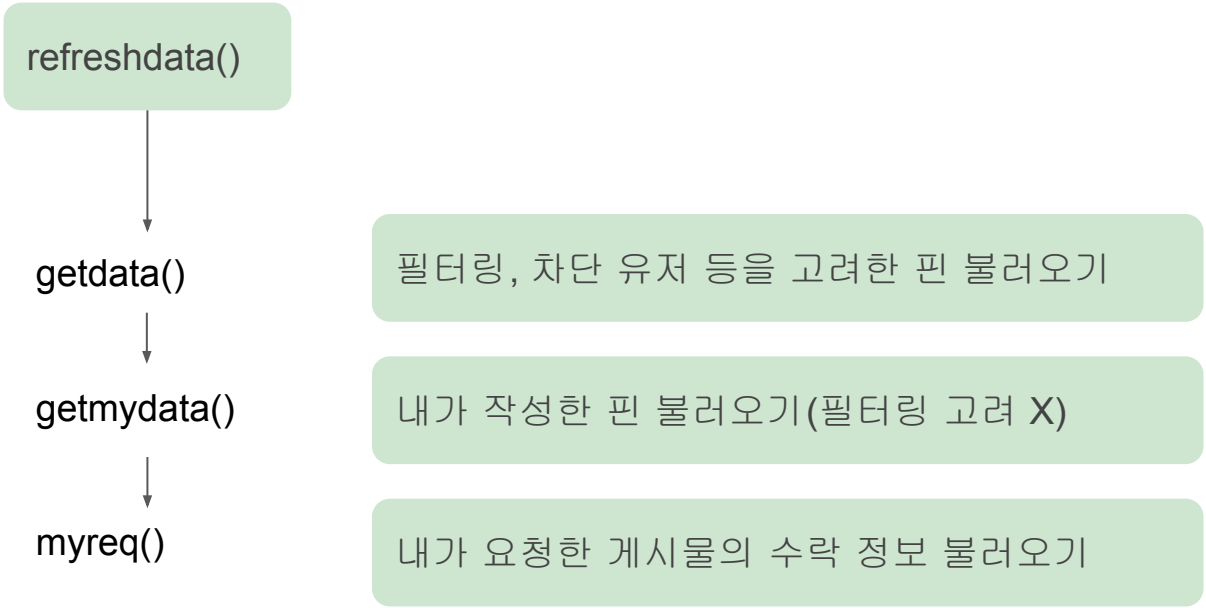
loadImageToBitmap()

일반 이미지와 작은 사이즈의 이미지로 각각 불러올 수 있다.

getResizedImage()

이미지가 너무 커서 로딩에 시간이 과하게 소모되는 것을 막기 위해 이미지 퀄리티가 MAX_IMAGE_SIZE 이하가 될 때까지 while문을 통해 저하시킨다.

Implementation - 산책 게시판(map)



Implementation - 산책 게시판(map)

getLastLocation()

위치 데이터의 최근 캐시 데이터를 이용해서 초기 위치를 설정한다.

mapsync()

onMapReady를 불러 map을 새로고침한다. getdata(), getmydata(), myreq()의 실행 마지막 단계에 모두 들어간다.

onMapReady()

핀이 중복 생성되는 것을 방지하기 위해 기존에 있던 핀은 모두 삭제한다. 요청 리스트에 대한 수락 여부를 체크하고 setCaption을 통해 마커 위에 표시한다.

Implementation - 산책 게시판(map)

마커를 클릭했을 때

getuserdata()를 통해 해당 마커의 작성자 정보를 로드한다. 이미지를 불러오는 데에 시간이 걸리기 때문에 로딩중 화면을 보여주고, 로딩이 끝났을 때 작성자 정보창을 보여준다.

메이트 신청 버튼을 클릭했을 때

checkandsendreq()를 통해 신청할 수 있는 게시물인지를 체크하고 요청을 전송한다. 이때 HashMap으로 변환하여 파이어스토어로 정보가 전송된다. 요청이 끝다면 refreshdata를 진행한다.

Back-End

Implementation - 채팅

Implementation - 채팅

1대1 만남

+

다수의 만남

Implementation - 채팅

1대1 만남

+

다수의 만남

FireBase 저장

+

Local 저장

Implementation - 채팅

ChatRoom

```
public class ChatRoom {
    public String roomid;
    public String roomname;
    public Map<String,Boolean> userids=new HashMap<~>();
    public Map<String,Comment> comments=new HashMap<>();

    public ChatRoom(){
    }

    public ChatRoom(String roomid, String roomname, Map<String, Boolean> userids) {
        this.roomid = roomid;
        this.roomname = roomname;
        this.userids = userids;
    }

    public static class Comment{
        public String msg;
        public String time;
        public String userid;

        public Comment(){}
    }

}
```

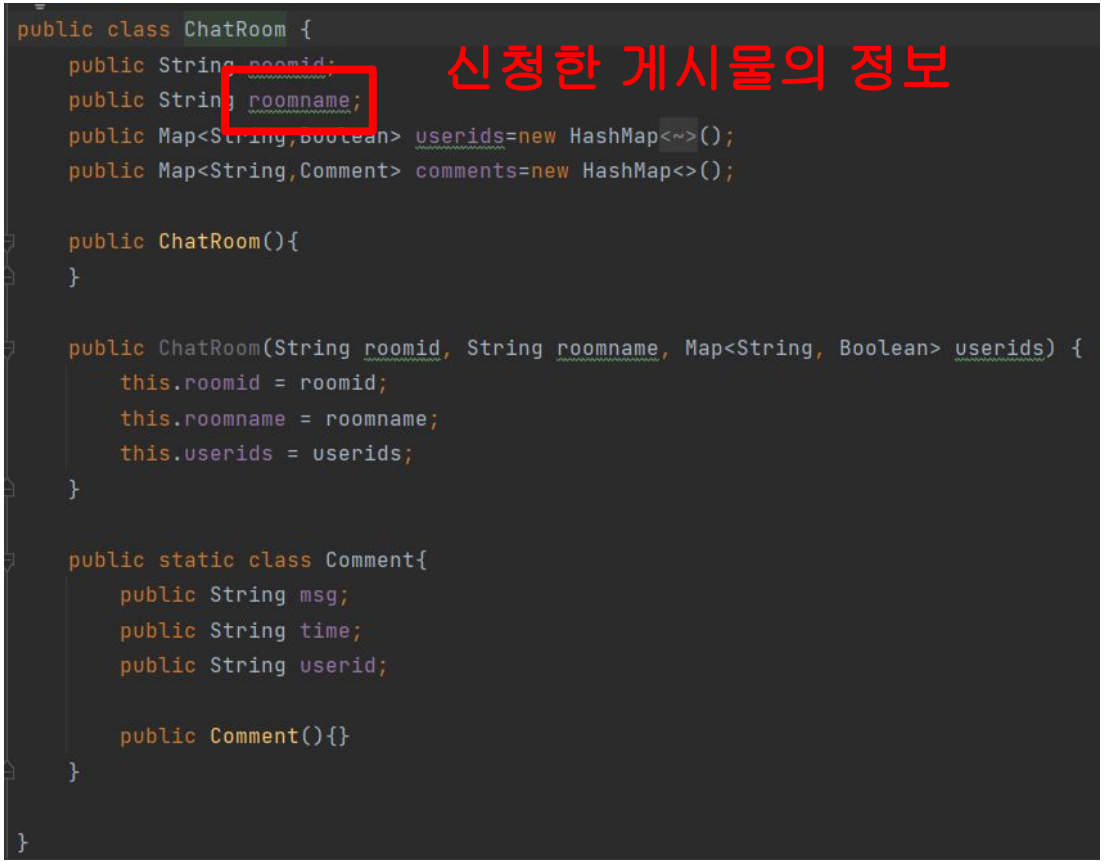
Implementation - 채팅

ChatRoom

```
public class ChatRoom {
    public String roomid;
    public String roomname;
    public Map<String,Boolean> userids=new HashMap<~>();
    public Map<String,Comment> comments=new HashMap<>();

    public ChatRoom(){
    }

    public ChatRoom(String roomid, String roomname, Map<String, Boolean> userids) {
        this.roomid = roomid;
        this.roomname = roomname;
        this.userids = userids;
    }

    public static class Comment{
        public String msg;
        public String time;
        public String userid;

        public Comment(){}
    }

}
```

Implementation - 채팅

ChatRoom



Implementation - 채팅

ChatRoom

```
public class ChatRoom {
    public String roomid;
    public String roomname;
    public Map<String,Boolean> userids=new HashMap<~>();
    public Map<String,Comment> comments=new HashMap<>();

    public ChatRoom(){
    }

    public ChatRoom(String roomid, String roomname, Map<String, Boolean> userids) {
        this.roomid = roomid;
        this.roomname = roomname;
        this.userids = userids;
    }

    public static class Comment{
        public String msg;
        public String time;
        public String userid;

        public Comment(){}
    }

}
```

Implementation - 채팅

ChatRoom

```java
public class ChatRoom {
    public String roomid;
    public String roomname;
    public Map<String,Boolean> userids=new
    public Map<String,Comment> comments=new

    public ChatRoom(){
    }

    public ChatRoom(String roomid, String roomname, Map<String, Boolean> userids) {
        this.roomid = roomid;
        this.roomname = roomname;
        this.userids = userids;
    }

    public static class Comment{
        public String msg;
        public String time;
        public String userid;

        public Comment(){}
    }

}
```

```
userids
    C7VynmLzbvX9yXxViYZZxMQQpqeASDbQKg6XFuAniv: false
    ob_ua6RyFxqm66pBjej9gJ0VDyatPHLDu81RRis__xY: false
    qy-LTaqG2gfFXY3JbNJmRVInup3ensNQBhEBjPou-D: true
```

채팅방을 나간 **user**

채팅방에 있는 **user**

Implementation - 채팅

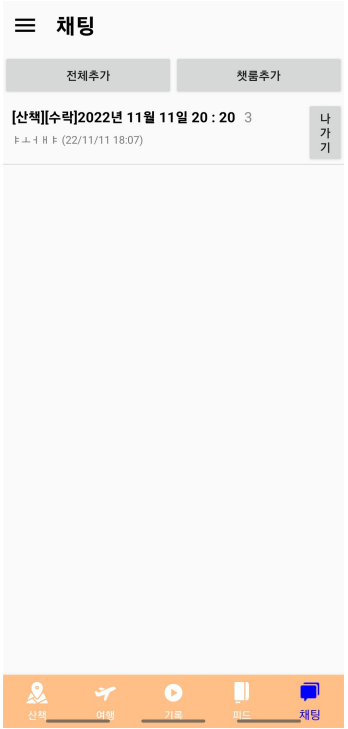
ChatRoom

```
public class ChatRoom {
    public String roomid;
    public String roomname;
    public Map<String,Boolean> userids=new HashMap<~>();
    public Map<String,Comment> comments=new HashMap<>();

    public ChatRoom(){
    }

    public ChatRoom(String roomid, String roomname, Map<String, Boolean> userids) {
        this.roomid = roomid;
        this.roomname = roomname;
        this.userids = userids;
    }

    public static class Comment{
        public String msg;
        public String time;
        public String userid;

        public Comment(){}
    }

}
```

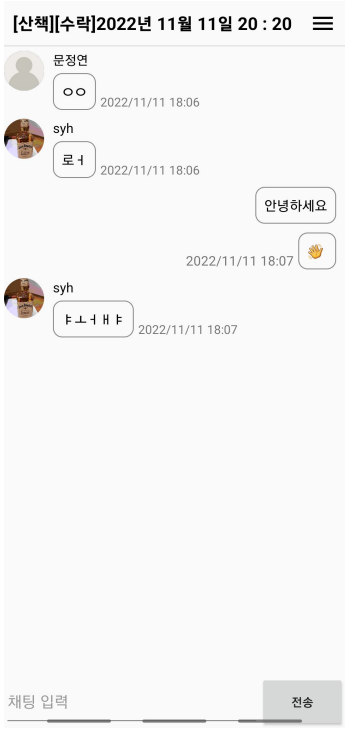
Back-End

Implementation - 채팅

ChatFragment



ChatActivity

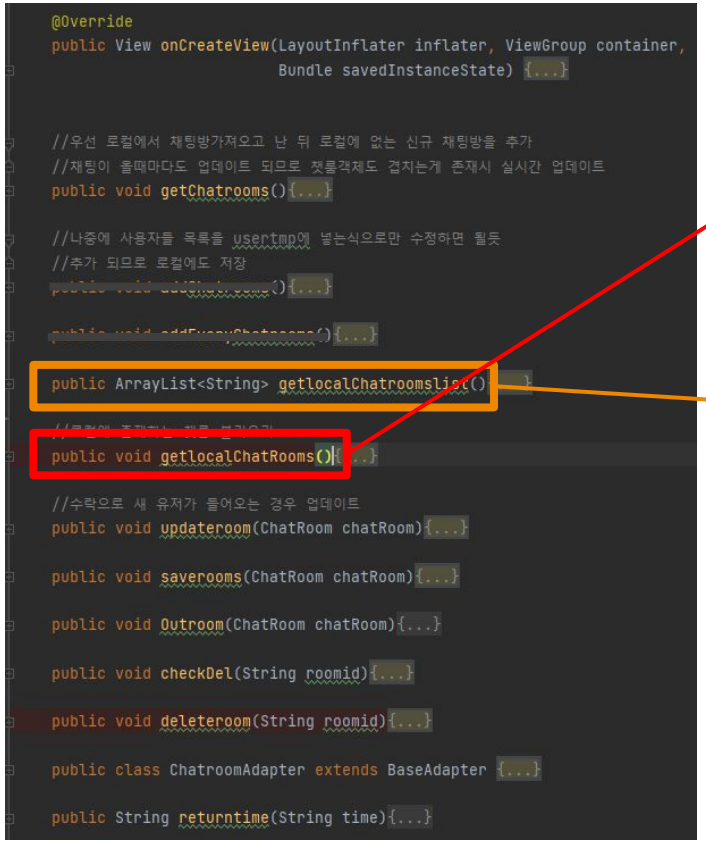


Implementation - 채팅

ChatFragment

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {...}


//우선 로컬에서 채팅방가져오고 난 뒤 로컬에 없는 신규 채팅방을 추가
//채팅이 올때마다도 업데이트 되므로 챗룸객체도 겹치는게 존재시 실시간 업데이트
public void getChatrooms(){...}

//나중에 사용자를 목록을 usertmp에 넣는식으로만 수정하면 될듯
//추가 되므로 로컬에도 저장
public void addChatrooms(){...}

public void addEveryChatrooms(){...}

public ArrayList<String> getlocalChatroomslist(){...}

//로컬에 존재하는 챗룸 불러오기
public void getlocalChatRooms(){...}

//수락으로 새 유저가 들어오는 경우 업데이트
public void updateroom(ChatRoom chatRoom){...}

public void saverooms(ChatRoom chatRoom){...}

public void Outroom(ChatRoom chatRoom){...}

public void checkDel(String roomid){...}

public void deleteroom(String roomid){...}

public class ChatroomAdapter extends BaseAdapter {...}

public String returntime(String time){...}
```

Implementation - 채팅



```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {...}


//우선 로컬에서 채팅방가져오고 난 뒤 로컬에 없는 신규 채팅방을 추가
//채팅이 올때마다도 업데이트 되므로 챗룸객체도 겹치는게 존재시 실시간 업데이트
public void getChatrooms(){...}

//나중에 사용자를 목록을 usertmp에 넣는식으로만 수정하면 될듯
//추가 되므로 로컬에도 저장
public void addChatrooms(){...}

public void addEveryChatrooms(){...}

public ArrayList<String> getlocalChatroomslist(){...}

//로컬에 존재하는 챗룸 불러오기
public void getlocalChatRooms(){...}

//수락으로 새 유저가 들어오는 경우 업데이트
public void updateroom(ChatRoom chatRoom){...}

public void saverooms(ChatRoom chatRoom){...}

public void Outroom(ChatRoom chatRoom){...}

public void checkDel(String roomid){...}

public void deleteroom(String roomid){...}

public class ChatroomAdapter extends BaseAdapter {...}

public String returntime(String time){...}
```

getlocalChatRooms

로컬에 저장된 채팅방 정보를 저장하고
리스트 띄운다.

getlocalChatroomslist

로컬에 저장된
채팅방 파일명 리스트를 불러온다.

Implementation - 채팅

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {...}


//우선 로컬에서 채팅방가져오고 난 뒤 로컬에 없는 신규 채팅방을 추가
//... 챗룸객체도 겹치는게 존재시 실시간 업데이트
public void getChatrooms(){...}

//나중에 사용자들 목록을 usertmp에 넣는식으로만 수정하면 될듯
//추가 되므로 로컬에도 저장
[redacted](){...}

[redacted](){...}

public ArrayList<String> getlocalChatroomslist(){...}

//로컬에 존재하는 챗룸 불러오기
public void getlocalChatRooms(){...}

public void updateroom(ChatRoom chatRoom){...}

public void saverooms(ChatRoom chatRoom){...}

public void Outroom(ChatRoom chatRoom){...}

public void checkDel(String roomid){...}

public void deleteroom(String roomid){...}

public class ChatroomAdapter extends BaseAdapter {...}

public String returntime(String time){...}
```

getChatrooms

파이어베이스 서버로부터
실시간으로 채팅룸 데이터를 받아온다.

updateroom

새로운 유저가 채팅방에 참가한 경우
채팅룸 정보가 담긴 파일을 변경

saverooms

기존에 존재하지 않는 채팅룸 데이터가 들어온 경우
로컬에 저장

Implementation - 채팅

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {...}


//우선 로컬에서 채팅방가져오고 난 뒤 로컬에 없는 신규 채팅방을 추가
//채팅이 올때마다도 업데이트 되므로 챗룸객체도 겹치는게 존재시 실시간 업데이트
public void getChatrooms(){...}

//나중에 사용자를 목록을 usertmp에 넣는식으로만 수정하면 될듯
//추가 되므로 로컬에도 저장
public void addChatrooms(){...}

public void addEveryChatrooms(){...}

public ArrayList<String> getlocalChatroomslist(){...}

//로컬에 존재하는 챗룸 불러오기
public void getlocalChatRooms(){...}

//수락으로 새 유저가 들어오는 경우 업데이트
public void updateroom(ChatRoom chatRoom){...}

public void saverooms(ChatRoom chatRoom){...}

public void Outroom(ChatRoom chatRoom){...}

public void checkDel(String roomid){...}

public void deleteroom(String roomid){...}

public class ChatroomAdapter extends BaseAdapter {...}

public String returntime(String time){...}
```

outroom

User가 채팅방을 나갈 때

checkDel

참가유저를 체크해 전부 나간 상태인지 확인후 전부 나간 상태면 **realtime database**에서 채팅방 삭제.

deleteroom

로컬에서 채팅룸정보와 메시지파일 모두 삭제.

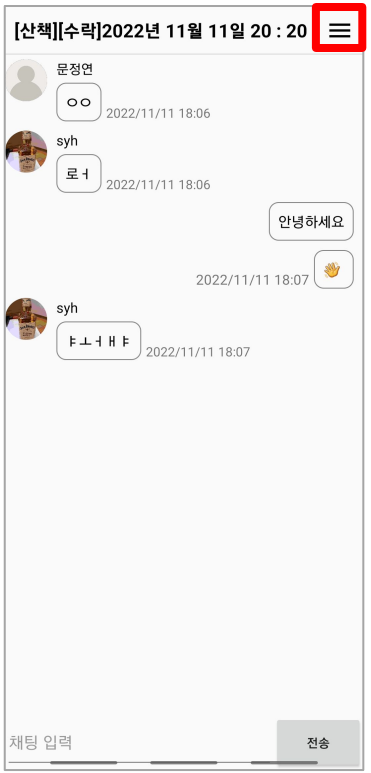
Implementation - 채팅

ChatActivity

```
public void settitle(String roomid){...}

//로컬에서 메시지 불러오기
//저장된 메시지의 마지막꺼를 starttime으로
public ArrayList<ChatRoom.Comment> setChattingroom(String roomid){...}


public void sendmsgs(){...}

public void savemsg(String msg){...}

public void checkuser(){...}

//제일 먼저 로컬에서 사진파일 불러오기
public void loadlocalprofile(){...}

public void savelocalprofile(Bitmap bitmap,String userid){...}

//hashmap에 있는지 체크. 없으면 불러오기 시작
//첫 시작, 새유저 접속시 작동
//목록에 없으면 저장
public void loaduserprofile(){...}

public String gettime(String time){...}


public class MsgAdapter extends BaseAdapter {...}

public class UserAdapter extends BaseAdapter {...}
```

Implementation - 채팅

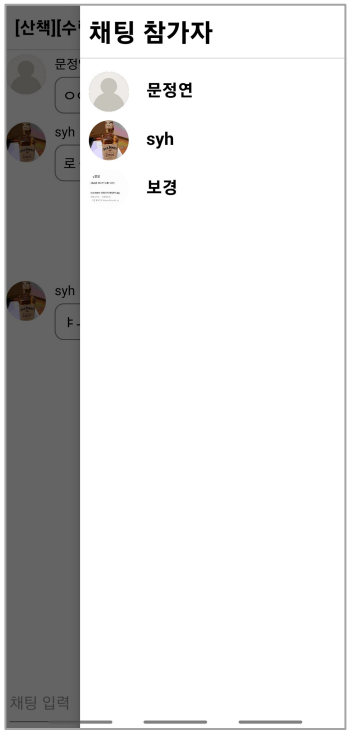
```
public void settitle(String roomid){...}

//로컬에서 메시지 불러오기
//저장된 메시지의 마지막꺼를 starttime으로
public ArrayList<ChatRoom.Comment> setChattingroom(String roomid){...}


public void sendmsgs(){...}

public void savemsg(String msg){...}

public void checkuser(){...}

//제일 먼저 로컬에서 사진파일 불러오기
public void loadlocalprofile(){...}

public void savelocalprofile(Bitmap bitmap,String userid){...}

//hashmap에 있는지 체크. 없으면 불러오기 시작
//첫 시작, 새유저 접속시 작동
//목록에 없으면 저장
public void loaduserprofile(){...}

public String gettime(String time){...}


public class MsgAdapter extends BaseAdapter {...}

public class UserAdapter extends BaseAdapter {...}
```



Implementation - 채팅

```
public void settitle(String roomid){...}

//로컬에서 메시지 불러오기
//저장된 메시지의 마지막꺼를 starttime으로
public ArrayList<ChatRoom.Comment> setChattingroom(String roomid){...}


public void sendmsgs(){...}

public void savemsg(String msg){...}

public void checkuser(){...}

//제일 먼저 로컬에서 사진파일 불러오기
public void loadlocalprofile(){...}

public void savelocalprofile(Bitmap bitmap,String userid){...}

//hashmap에 있는지 체크. 없으면 불러오기 시작
//첫 시작, 새유저 접속시 작동
//목록에 없으면 저장
public void loaduserprofile(){...}

public String gettime(String time){...}


public class MsgAdapter extends BaseAdapter {...}

public class UserAdapter extends BaseAdapter {...}
```

UserAdapter



Side Bar User List

DrawerLayout

⬇️

ListView

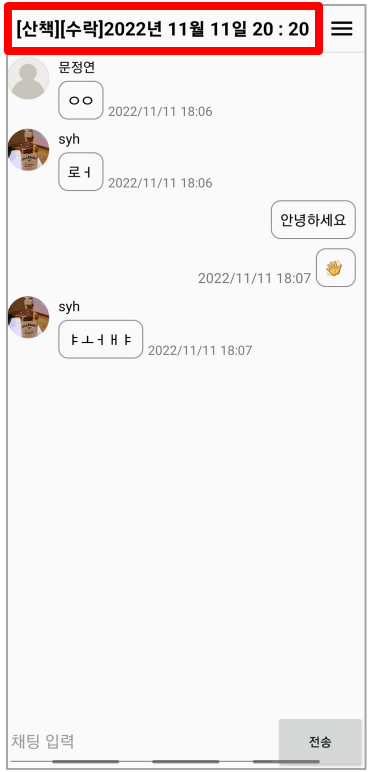
2 Implementation - 채팅

```
public void settitle(String roomid) {...}

//로컬에서 메시지 불러오기
//저장된 메시지의 마지막꺼를 starttime으로
public ArrayList<ChatRoom.Comment> setChattingroom(String roomid){...}


public void sendmsgs(){...}

public void savemsg(String msg){...}

public void checkuser(){...}

//제일 먼저 로컬에서 사진파일 불러오기
public void loadlocalprofile(){...}

public void savelocalprofile(Bitmap bitmap,String userid){...}

//hashmap에 있는지 체크. 없으면 불러오기 시작
//첫 시작, 새유저 접속시 작동
//목록에 없으면 저장
public void loaduserprofile(){...}

public String gettime(String time){...}


public class MsgAdapter extends BaseAdapter {...}

public class UserAdapter extends BaseAdapter {...}
```

settitle

- **DB에서 roomname을 받아와서 상단에 띄워준다.**

- **User가 채팅룸에 처음 접근할 때만 사용.**



Implementation - 채팅

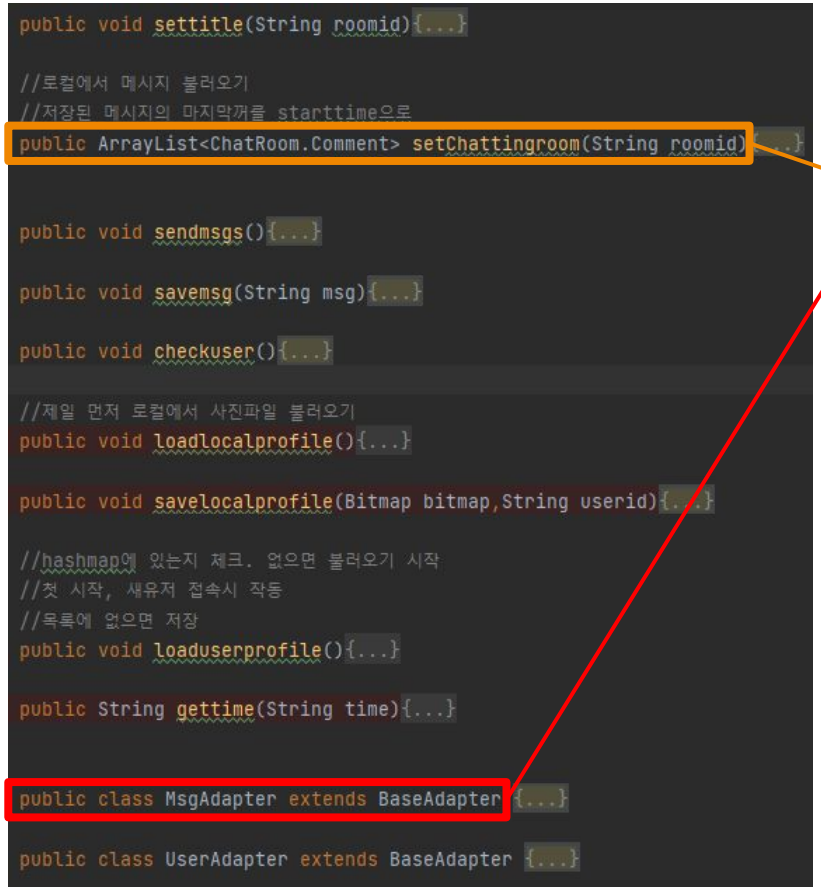
```
public void settitle(String roomid){...}

//로컬에서 메시지 불러오기
//저장된 메시지의 마지막꺼를 starttime으로
public ArrayList<ChatRoom.Comment> setChattingroom(String roomid){...}


public void sendmsgs(){...}

public void savemsg(String msg){...}

public void checkuser(){...}

//제일 먼저 로컬에서 사진파일 불러오기
public void loadlocalprofile(){...}

public void savelocalprofile(Bitmap bitmap,String userid){...}

//hashmap에 있는지 체크. 없으면 불러오기 시작
//첫 시작, 새유저 접속시 작동
//목록에 없으면 저장
public void loaduserprofile(){...}

public String gettime(String time){...}


public class MsgAdapter extends BaseAdapter {...}

public class UserAdapter extends BaseAdapter {...}
```

checkuser

- 실시간으로 채팅방 참가자 리스트를 체크한다.

- 새로운 참가자가 있으면 **DB**에서 데이터를 받아온다.

Implementation - 채팅



```
public void settitle(String roomid){...}

//로컬에서 메시지 불러오기
//저장된 메시지의 마지막꺼를 starttime으로
public ArrayList<ChatRoom.Comment> setChattingroom(String roomid){...}


public void sendmsgs(){...}

public void savemsg(String msg){...}

public void checkuser(){...}

//제일 먼저 로컬에서 사진파일 불러오기
public void loadlocalprofile(){...}

public void savelocalprofile(Bitmap bitmap,String userid){...}

//hashmap에 있는지 체크. 없으면 불러오기 시작
//첫 시작, 새유저 접속시 작동
//목록에 없으면 저장
public void loaduserprofile(){...}

public String gettime(String time){...}


public class MsgAdapter extends BaseAdapter {...}

public class UserAdapter extends BaseAdapter {...}
```

MsgAdapter

메시지 띄우기.

setChattingroom

- 로컬에 저장된 메시지 파일을 읽어서 **comments** 객체로 만든다.
- **comments**를 **ArrayList**로 반환한다.
- 로컬에 저장된 메시지 중 가장 최신 것의 타임스탬프를 **‘startime’**에 저장한다.

Implementation - 채팅

```
public class MsgAdapter extends BaseAdapter {

    LayoutInflater layoutInflater;
    Context context;
    ArrayList<ChatRoom.Comment> comments=new ArrayList<>();
    String roomid;
    String userid;

    public MsgAdapter(Context context, String roomid, String userid){...}

    @Override
    public int getCount() { return comments.size(); }

    @Override
    public Object getItem(int i) { return null; }

    @Override
    public long getItemId(int i) { return i; }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {...}


    //이건 추가된것만 가져오므로 선택
    //startAt으로 기존 것은 안가져오게 설정
    //메시지는 로컬에 저장해두기
    public void getrecentmsg(){...}
}
```

MsgAdapter

메시지 띄우기.

setChattingroom

- 로컬에 저장된 메시지 파일을 읽어서 **comments** 객체로 만든다.
- **comments**를 **ArrayList**로 반환한다.
- 로컬에 저장된 메시지 중 가장 최신 것의 타임스탬프를 **'startime'**에 저장한다.

getrecentmsg

- 새로운 메시지를 **'savemsg'** 메소드를 통해 로컬에 저장한다.
- **'starttime'** 이후의 메시지만 불러온다.

Implementation - 채팅

```
public void settitle(String roomid){...}

//로컬에서 메시지 불러오기
//저장된 메시지의 마지막꺼를 starttime으로
public ArrayList<ChatRoom.Comment> setChattingroom(String roomid){...}


public void sendmsgs(){...}

public void savemsg(String msg){...}

public void checkuser(){...}

//제일 먼저 로컬에서 사진파일 불러오기
public void loadlocalprofile(){...}

public void savelocalprofile(Bitmap bitmap,String userid){...}

//hashmap에 있는지 체크. 없으면 불러오기 시작
//첫 시작, 새유저 접속시 작동
//목록에 없으면 저장
public void loaduserprofile(){...}

public String gettime(String time){...}


public class MsgAdapter extends BaseAdapter {...}

public class UserAdapter extends BaseAdapter {...}
```

sendmsgs

- **Activity**에서 메시지 작성 후 전송 버튼을 누르면 실행.

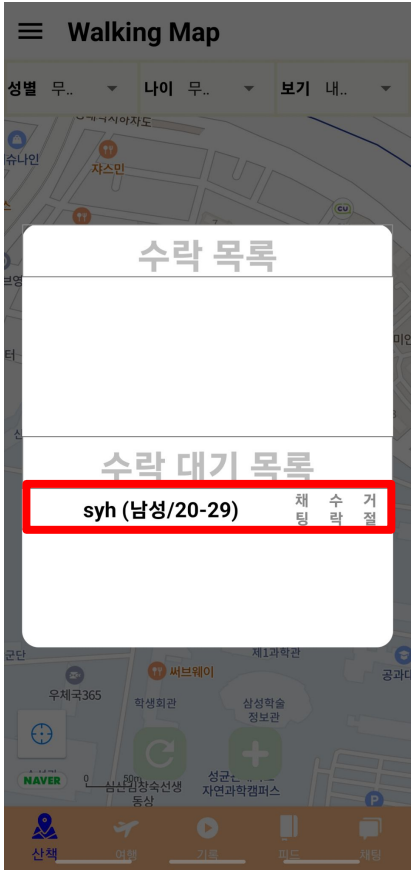
- 메시지, **timestamp, userid**를 **DB**에 올린다.

Implementation - 채팅

1. 메이트를 신청한다.
2. 게시자에게 신청이 들어간다.
3. 게시자가 수락 전 1대1 채팅방을 생성 가능
4. 게시자가 신청을 수락하면
   수락채팅방이 생성된다. (1대1 채팅방은 유지)
5. 다른 유저가 수락되면 수락 채팅방에 user를 추가

Implementation - 채팅

1. 메이트를 신청한다.
2. **게시자에게 신청이 들어간다.**
3. 게시자가 수락 전 1대1 채팅방을 생성 가능
4. 게시자가 신청을 수락하면
   수락채팅방이 생성된다. (1대1 채팅방은 유지)
5. 다른 유저가 수락되면 수락 채팅방에 user를 추가

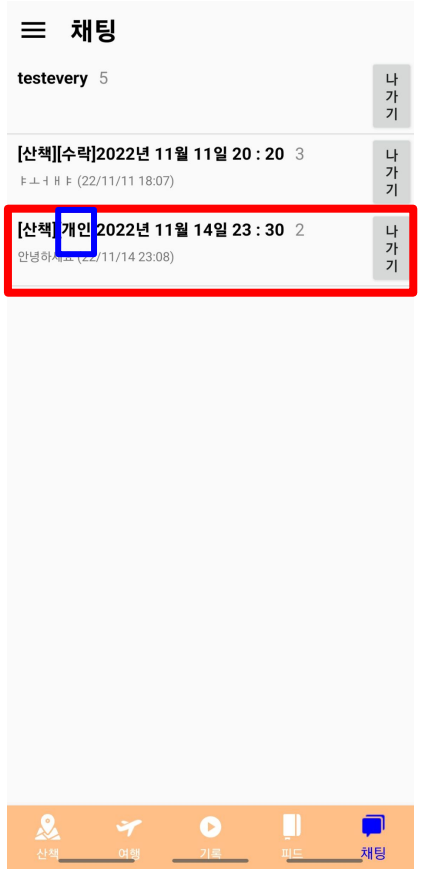


Implementation - 채팅

1. 메이트를 신청한다.
2. 게시자에게 신청이 들어간다.
3. **게시자가 수락 전 1대1 채팅방을 생성 가능**
4. 게시자가 신청을 수락하면
   수락채팅방이 생성된다. (1대1 채팅방은 유지)
5. 다른 유저가 수락되면 수락 채팅방에 user를 추가

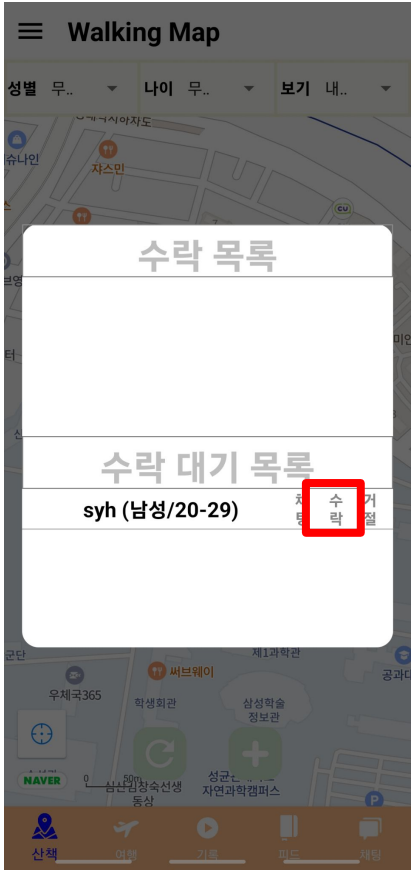
Implementation - 채팅

1. 메이트를 신청한다.
2. 게시자에게 신청이 들어간다.
3. **게시자가 수락 전 1대1 채팅방을 생성 가능**
4. 게시자가 신청을 수락하면
   수락채팅방이 생성된다. (1대1 채팅방은 유지)
5. 다른 유저가 수락되면 수락 채팅방에 user를 추가



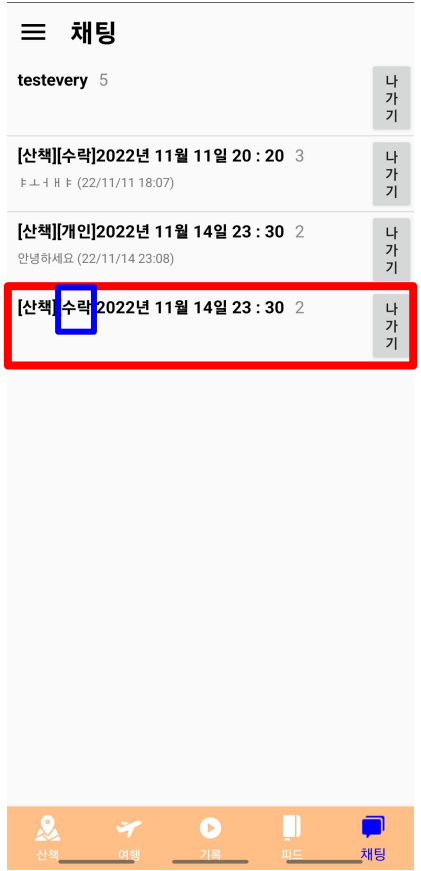
Implementation - 채팅

1. 메이트를 신청한다.
2. 게시자에게 신청이 들어간다.
3. 게시자가 수락 전 1대1 채팅방을 생성 가능
4. **게시자가 신청을 수락하면**
   수락채팅방이 생성된다. (1대1 채팅방은 유지)
5. 다른 유저가 수락되면 수락 채팅방에 user를 추가



Implementation - 채팅

1. 메이트를 신청한다.
2. 게시자에게 신청이 들어간다.
3. 게시자가 수락 전 1대1 채팅방을 생성 가능
4. **게시자가 신청을 수락하면**
   수락채팅방이 생성된다. (1대1 채팅방은 유지)
5. 다른 유저가 수락되면 수락 채팅방에 user를 추가



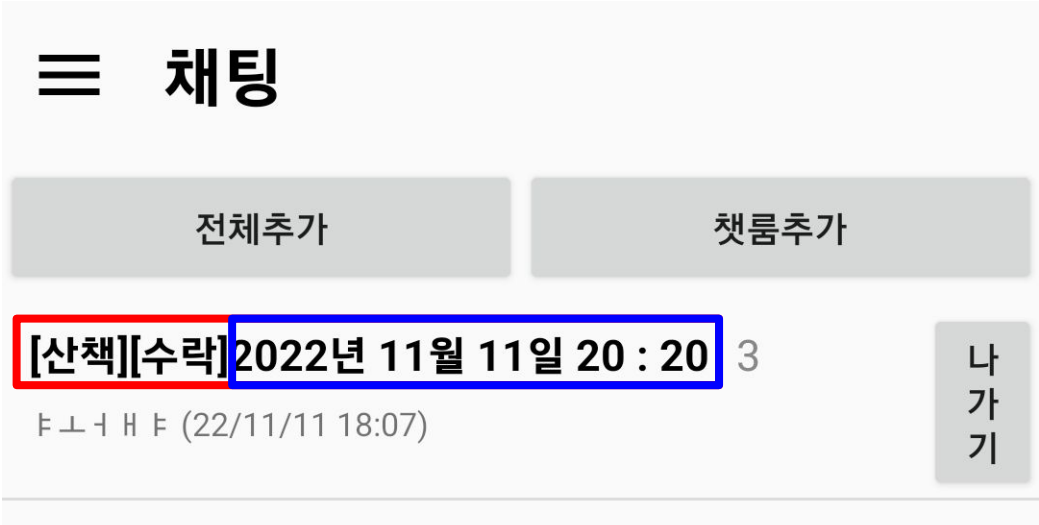
Implementation - 채팅

1. 메이트를 신청한다.
2. 게시자에게 신청이 들어간다.
3. 게시자가 수락 전 1대1 채팅방을 생성 가능
4. 게시자가 신청을 수락하면
   수락채팅방이 생성된다. (1대1 채팅방은 유지)
5. **다른 유저가 수락되면 수락 채팅방에 user를 추가**

Implementation - 채팅

Implementation - 채팅

Implementation - 채팅

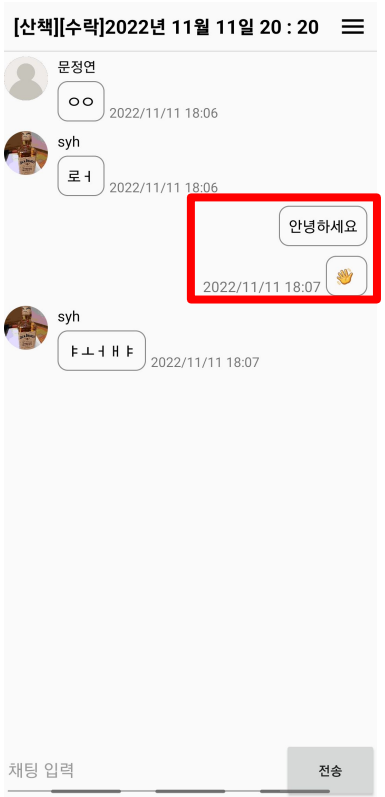


Implementation - 채팅

Implementation - 채팅

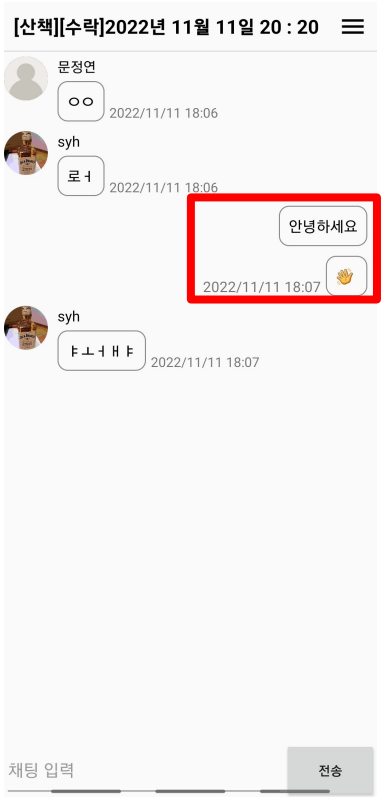
Back-End

Implementation - 채팅



Back-End

Implementation - 채팅



Next Week

3 Next Week

Front - End

1. 유저 프로필 페이지 구현
2. 여행게시판 구현
3. 약속 목록 및 리뷰 구현

Back - End

1. 파이어베이스 보안규칙 적용
2. 버그테스트

Thank you!