Faculty of Computer Science and Engineering

Ho Chi Minh City University of Technology



**CHAT LOCAL PROJECT**

**DETAIL DESIGN DOCUMENT**

**Class Design**

**Project Code:32**

**Document Code: 32-DD – v1.0**

**Bach Khoa, 30/3/2018**

**RECORD OF CHANGE**

\*A - Added M - Modified D – Deleted

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Effective Date** | **Changed Items** | **A\* M, D** | **Change Description** | **New Version** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **7/4/2018** | **Add controller login class** | **Added** | **Class include all control with account login, logout, register** | **V1.0** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**SIGNATURE PAGE**

**ORIGINATOR:** Nguyễn Thành Công 30/3/2017 Leader

**REVIEWERS:** Nguyễn Thành Đạt 31/3/2017 Member

Nguyễn Trọng Tín 31/3/2017 Member

Nguyễn Đỗ Đức Anh 31/3/2017 Member

Hoàng Hùng 31/03/2018 Teacher Assistant

**APPROVAL:** Hoàng Hùng 31/03/2018 Teacher Assistant

**TABLE OF CONTENTS**

[1 Introduction 5](#_Toc517625043)

[1.1 Purpose 5](#_Toc517625044)

[1.2 Definitions, Acronyms and Abbreviations 5](#_Toc517625045)

[1.3 Standards and Conventions 5](#_Toc517625046)

[1.4 References 6](#_Toc517625047)

[1.5 Overview 6](#_Toc517625048)

[2 Chat Loca 7](#_Toc517625049)

[2.1 *Class diagram* 7](#_Toc517625050)

[2.2 *BaseActivity* 8](#_Toc517625051)

[2.3 *ChatActivity* 14](#_Toc517625052)

[2.4 *MapsActivity* 18](#_Toc517625053)

[2.5 *MessageActivity* 19](#_Toc517625054)

[2.6 *ProfileActivity* 39](#_Toc517625055)

[2.7 *Sign\_inActivity* 41](#_Toc517625056)

[2.8 *Sign\_upActivity* 42](#_Toc517625057)

[2.9 *ViewProfileActivity* 44](#_Toc517625058)

[2.10 *Message\_Adapter* 48](#_Toc517625059)

[2.11 *CreateGroupDialog* 49](#_Toc517625060)

[2.12 *SearchMemberDialog* 51](#_Toc517625061)

[3 Diagrams 60](#_Toc517625062)

[5 Packages 61](#_Toc517625063)

[6 Other considerations 62](#_Toc517625064)

[7 Appendix 62](#_Toc517625065)

# Introduction

## Purpose

Folder design The Chat Loca class contains the full description of all classes in the application system including:

The Class diagram describes the static relationships of all classes in the system

Dynamic diagrams, such as a collaboration diagram, sequence diagram, activity diagram

Detailed description of each layer and the implementation of each layer

The developer and tester will rely on this design to implement class and unit tests

## Definitions, Acronyms and Abbreviations

|  |  |  |
| --- | --- | --- |
| **Abbreviations** | **Description** | **Comment** |
| SDK | Software Development Kit | typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform. |
| DB | Database | Is computer-software application that interacts with end-users, other applications, and the database itself to capture and analyze data. |

## Standards and Conventions

*Define all the conventions to write class design:*

*Design standards: LibreOffice Writer + Word Office Microsoft*

*Documentation standards:*

*Font: Arial*

*Color: Black #000000*

*Style: This page intentionally left blank*

*Formal style:*

*In this document, we use java convention to easier to read yourself and for other programmers.*

|  |  |
| --- | --- |
| **Name** | **Convention** |
| class name | should start with uppercase letter and be a noun e.g. Controler, Login etc. |
| interface name | should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc. |
| constants name | should be in uppercase letter. e.g. RED, YELLOW, MAX\_PRIORITY etc. |
| method name | should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc. |
| variable name | should start with lowercase letter e.g. firstName, orderNumber etc. |
| package name | should be in lowercase letter e.g. java, lang, sql, util etc. |

## References

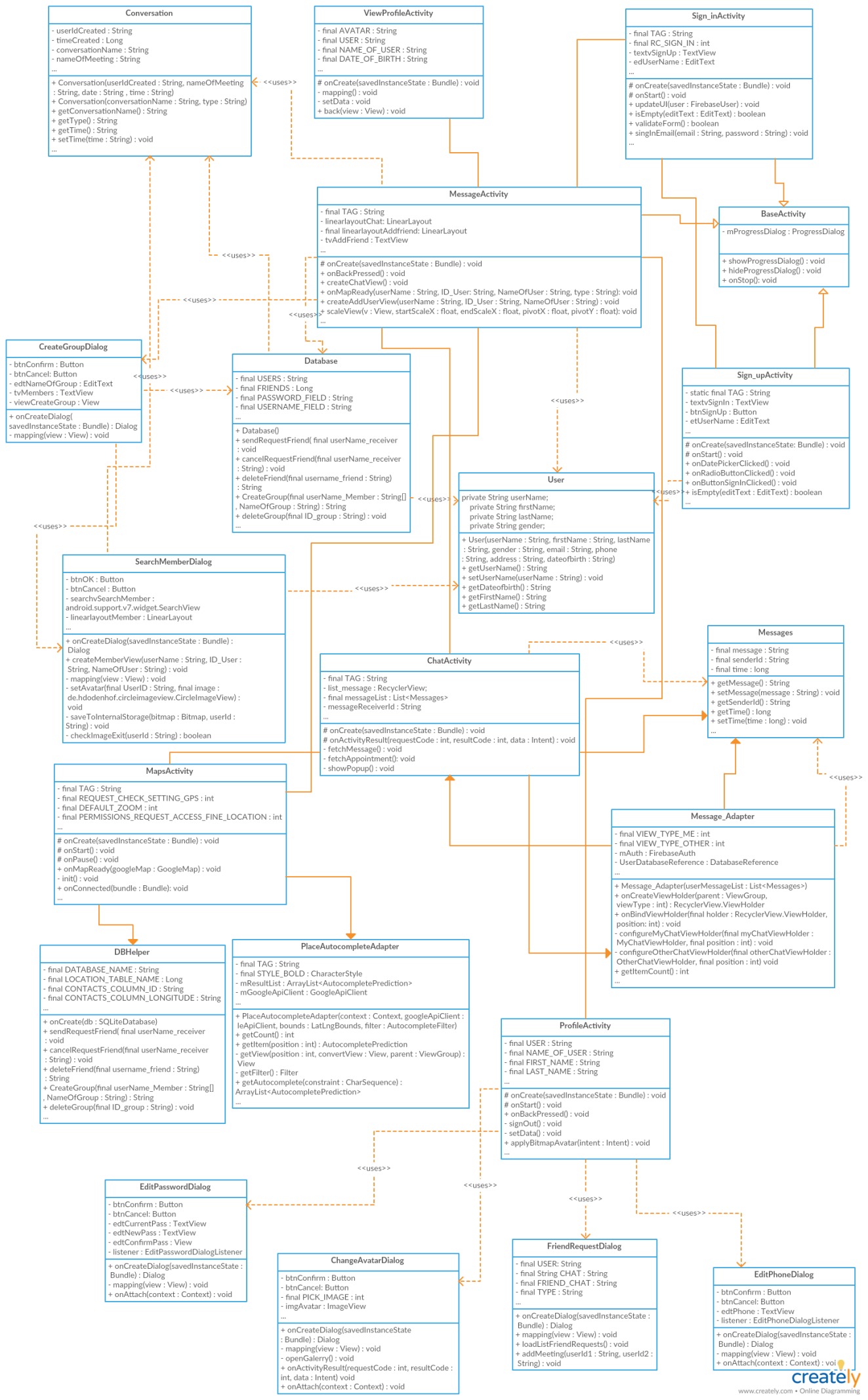
*<List all the reference document such as: other document of the system, or the technical article,...>*

|  |  |
| --- | --- |
| **Document Number** | **Title** |
| 01 | Data design document |
|  |  |

## Overview

*<General overview of the detail design such as what is the structure of the document>*

# Chat Loca

* 1. ***Class diagram***

|  |  |  |
| --- | --- | --- |
| **No** | **Class Name** | **Description** |
| 01 | BaseActivity | This class present for base activites related with screen, keybroad |
| 02 | ChatActivity | This class present for Chat Acitvity |
| 03 | MapsActivity | This class present for Map Activity |
| 04 | MessageActivity | This class present for Message Activity |
| 05 | ProfileActivity | This class present for Profile Activity |
| 06 | Sign\_inActivity | This class present for Sign in Activity |
| 07 | Sign\_upActivity | This class present for Sign up Activity |
| 08 | ViewProfileActivity | This class present for View profile Activity |
| 09 | Message\_Adapter | This class present for message in Chat view |
| 10 | PlaceAutocompleteAdapter | This class present for Place Auto complete Adapter |
| 11 | ChangeAvatarDialog | This class present for Change Avatar Dialog |
| 12 | CreateGroupDialog | This class present for Dialog of creating group chat |
| 13 | EditPasswordDialog | This class present for Edit Password Dialog |
| 14 | EditPhoneDialog | This class present for Edit Phone Dialog |
| 15 | FriendRequestDialog | This class present for Friend Request Dialog |
| 16 | SearchMemberDiaglog | This class present for Dialog of searching friend to pick members for group |
| 17 | Conversation | This class present for conversation constructor |
| 18 | Database | This class present for action related firebase |
| 19 | Messages | This class present for Message constructor |
| 20 | User | This class present for User constructor |

* 1. ***BaseActivity***

This class present for base activites related with screen, keybroad

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | mProgressDialog | ProgressDialog |  | Private |  |

* **Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | showProgressDialog | Show Progress dialog |
| 02 | hideProgressDialog | Hide progress dialog |
| 03 | hideKeyboard | Hide keybroad |

**01. showProgressDialog method**

*public void showProgressDialog()*

Description: Show progress dialog

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

***public void*** *showProgressDialog() {*

**if** (**mProgressDialog** == **null**) {

**mProgressDialog** = **new** ProgressDialog(**this**);

**mProgressDialog**.setMessage(getString(R.string.***loading***));

**mProgressDialog**.setIndeterminate(**true**);

}

**mProgressDialog**.show();

}

**02. hideProgressDialog method**

*public void hideProgressDialog()*

Description: hide progress dialog

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

**public void** hideProgressDialog() {

**if (mProgressDialog != null && mProgressDialog.isShowing()) {**

**mProgressDialog.dismiss();**

**}**

**}**

**03. hideKeyboard method**

*public void hideKeyboard()*

Description: hide keyboard

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

**public void hideKeyBoard**() {

**final** InputMethodManager imm = (InputMethodManager) getSystemService(Context.***INPUT\_METHOD\_SERVICE***);

**if** (imm != **null**) {

imm.hideSoftInputFromWindow(view.getWindowToken(), 0);

}

}

* 1. ***ChatActivity***

This class present for Chat Activity.

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | TAG | String | "ChatActivity" | Private | the final variable |
| 02 | list\_message | RecyclerView |  | Private | recycler consist of messages |
| 03 | messageList | List<Messages> |  | Private | list message of the conversation |
| 04 | messageReceiverId | String |  | Private | receiver's messase's ID |
| 05 | messageReceiverName | String |  | Private | friend's name or group's name |
| 06 | messageSenderId | String |  | Private | sender's messase's ID |
| 07 | typeChat | String |  | Private | type of chat |
| 08 | address | String |  | Private | address of the meeting |
| 09 | longitude | Double |  | Private | longitude of position on the map of the meeting |
| 10 | latitude | Double |  | Private | latitude of position on the map of the meeting |
| 11 | MY\_REQUEST\_CODE | int | 100 | Public | to get response from intent |
| 12 | rootRef | DatabaseReference |  | Private | root of firebase |
| 13 | mAuth | FirebaseAuth |  | Private | the authentication of user |
| 14 | linearLayoutManager | LinearLayoutManager |  | Private | layout manager of list message |
| 15 | messageAdapter | Message\_Adapter |  | Private | handle message view |
| 16 | btnSendMessage | ImageView |  | Private | button send message |
| 17 | editText | EditText |  | Private | to enter message's content |
| 18 | myDialog | Dialog |  | Private |  |
| 19 | btnSetApp | Button |  | Private |  |
| 20 | btnSetPos | Button |  | Private |  |
| 21 | btnDate | Button |  | Private |  |
| 22 | btnTime | Button |  | Private |  |
| 23 | txtDate | TextView |  | Private |  |
| 24 | txtTime | TextView |  | Private |  |
| 25 | edtName | EditText |  | Private |  |
| 26 | mYear | int |  | Private |  |
| 27 | mMonth | int |  | Private |  |
| 28 | mDay | int |  | Private |  |
| 29 | mHour | int |  | Private |  |
| 30 | mMinute | int |  | Private |  |
| 31 | txt\_name | TextView |  | Private |  |
| 32 | txt\_date | TextView |  | Private |  |
| 33 | txt\_time | TextView |  | Private |  |
| 34 | txtNameLocation | TextView |  | Private |  |
| 35 | name | String |  | Private |  |
| 36 | date | String |  | Private |  |
| 37 | time | String |  | Private |  |
| 38 | goBack | Button |  | Private |  |

* **Methods**

|  |  |  |
| --- | --- | --- |
|  | **Method** | **Description** |
| 01 | getId | Get value of id |
| 02 | getMessages | Get value of listMessages |
| 03 | getApmts | Get value of listApmts |
| 04 | appendMsg | Append a message to listMessages |
| 05 | appendApmt | Append an appointment to listApmts |

**01. getId method**

*public String getId()*

Description: Get value of id

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | return id | String | out |  | return id private variable |

* **Implementation**

*return id*

**02. getMessages method**

*public String[] getMessages()*

Description: Get value of listMessages

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | return listMessages | String Array | out |  | return listMessages private variable |

* **Implementation**

*return listMessages*

**03. getApmts method**

*public String[] getApmts()*

Set value of listApmts

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | return listApmts | String Array | In |  | return listApmts private variable |

* **Implementation**

*return listApmts*

**04. appendMsg method**

*public void appendMsg (String msgId)*

Description: Append a message to listMessages

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | msgId | String | In |  | Message’s ID that we want to append |

* **Implementation**

*listMessages.append(msgId)*

**05. appendApmt method**

*public void appendApmt (String apmtId)*

Description: Append an appointment to listApmts

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | apmtId | String | In |  | Appointment’s ID that we want to append |

* **Implementation**

*listApmts.append(apmt)*

* 1. ***MapsActivity***

This class present for a maps activity.

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | mMap | GoogleMap |  | Private | Map instance |
| 02 | mLastKnownLocation | Location |  | Private | Store last known location of user |
| 03 | mConverId | String |  | Private |  |
| 04 | mMarkerMeeting | Marker |  | Private | Maker for meeting set up |

* **Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | onCreate | Initialize MapsActivity |
| 02 | getDeviceLocation | Get device Location and show in map |
| 04 | initMap | Initilize map |
| 05 | setRoute | Set route to Meeting location |
| 06 | onMarkerClick | Show marker meeting location |

**01. onCreate method**

*public void onCreate()*

Description: Initialize MapsActivity

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |

* **Implementation**

**super**.onCreate(savedInstanceState);

**if**(savedInstanceState != **null**){

**mLastKnownLocation** = savedInstanceState.getParcelable(***KEY\_LOCATION***);

**mCameraPosition** = savedInstanceState.getParcelable(***KEY\_CAMERA\_POSITION***);

**mLocationMeeting** = savedInstanceState.getParcelable(***KEY\_MEETING\_LOCATION***);

Log.*d*(***TAG***, **"GET THIS LOCATION "**+ **mLastKnownLocation**.getLongitude()+ **" "**+ **mLastKnownLocation**.getLatitude());

}

setContentView(R.layout.***activity\_maps***);

**polylines** = **new** ArrayList<>();

**btSetLocation** = findViewById(R.id.***Map\_btn\_set\_location***);

**btCancel** = findViewById(R.id.***Map\_btn\_set\_cancel***);

**mGps** = findViewById(R.id.***ic\_gps***);

**mSearchText** = findViewById(R.id.***input\_search***);

**mSearch** = findViewById(R.id.***ic\_magnify***);

**mRoute** = findViewById(R.id.***ic\_route***);

**mInfo** = findViewById(R.id.***meeting\_info***);

**mGoogleApiClient** = **new** GoogleApiClient.Builder(**this**)

.enableAutoManage(**this**, 0,**this** )

.addConnectionCallbacks(**this**)

.addApi(LocationServices.***API***)

.addApi(Places.***GEO\_DATA\_API***)

.addApi(Places.***PLACE\_DETECTION\_API***)

.build();

**mGoogleApiClient**.connect();

initMap();

getLocationPermission();

**02. getDeviceLocation method**

*public void getDeviceLocation()*

Description: Get device location

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |

* **Implementation**

**private void getDeviceLocation() {**

***/\****

***\* Get the best and most recent location of the device, which may be null in rare***

***\* cases when a location is not available.***

***\*/***

**Log.*d*(*TAG*, "onGetDeviceLocation");**

**try {**

**if (mLocationPermissionGranted) {**

**LocationRequest mlocationRequest = new LocationRequest();**

**mlocationRequest.setInterval(60 \* 60 \* 1000);**

**mlocationRequest.setFastestInterval(120 \* 1000);**

**mlocationRequest.setPriority(LocationRequest.*PRIORITY\_HIGH\_ACCURACY* );**

**LocationSettingsRequest.Builder builder = new LocationSettingsRequest.Builder()**

**.addLocationRequest(mlocationRequest);**

**builder.setAlwaysShow(true);**

**PendingResult result = LocationServices.*SettingsApi***

**.checkLocationSettings(mGoogleApiClient, builder.build());**

**result.setResultCallback(new ResultCallback() {**

**@Override**

**public void onResult(@NonNull Result result) {**

**final Status status = result.getStatus();**

**switch (status.getStatusCode()) {**

**case LocationSettingsStatusCodes.*SUCCESS*:**

**Log.*d*(*TAG*, "PERMISION: NO 1");**

***// All location settings are satisfied.***

***// You can initialize location requests here.***

**int permissionLocation = ContextCompat**

**.*checkSelfPermission*(MapsActivity.this,**

**android.Manifest.permission.*ACCESS\_FINE\_LOCATION*);**

**if (permissionLocation == PackageManager.*PERMISSION\_GRANTED*) {**

**final Task<Location> task= mFusedLocationProviderClient.getLastLocation();**

**task.addOnSuccessListener(MapsActivity.this, new OnSuccessListener<Location>() {**

**@Override**

**public void onSuccess(Location location) {**

**if(task.isSuccessful()){**

**Log.*d*(*TAG*, "get Current location SUCCESS");**

**mLastKnownLocation = task.getResult();**

**if(mLastKnownLocation !=null) {**

**Log.*d*(*TAG*,"mLASTKNOWN NOT EQUAL NULL");**

**mMap.moveCamera(CameraUpdateFactory.*newLatLngZoom*(new LatLng(mLastKnownLocation.getLatitude(),**

**mLastKnownLocation.getLongitude()), *DEFAULT\_ZOOM*));**

**}else**

**Log.*d*(*TAG*, "mLASTKNOWN IS EQUAL NULL");**

**}else{**

**Log.*d*(*TAG*, "Current location is null. Using defaults.");**

**Log.*e*(*TAG*, "Exception: %s", task.getException());**

**}**

**}**

**});**

**}**

**break;**

**case LocationSettingsStatusCodes.*RESOLUTION\_REQUIRED*:**

**Log.*d*(*TAG*, "PERMISION: NO 2");**

***// Location settings are not satisfied.***

***// But could be fixed by showing the user a dialog.***

**try {**

***// Show the dialog by calling startResolutionForResult(),***

***// and check the result in onActivityResult().***

***// Ask to turn on GPS automatically***

**status.startResolutionForResult(MapsActivity.this,**

***REQUEST\_CHECK\_SETTINGS\_GPS*);**

**} catch (IntentSender.SendIntentException e) {**

***// Ignore the error.***

**}**

**break;**

**case LocationSettingsStatusCodes.*SETTINGS\_CHANGE\_UNAVAILABLE*:**

**Log.*d*(*TAG*, "PERMISION: NO 3");**

***// Location settings are not satisfied. However, we have no way to fix the***

***// settings so we won't show the dialog.***

***//finish();***

***// to do: when not permission gps get db and show lastknowlocation***

**break;**

**}**

**}**

**});**

**}**

**else { *// to do: when not permission gps get db and show lastknowlocation***

**Log.*d*(*TAG*, "PERMISION: NULL");**

**}**

**} catch (SecurityException e) {**

**Log.*e*("Exception: %s", e.getMessage());**

**}**

**}**

**03. initMap method**

***public void*** *initMap()*

Description: Initialize Map

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

**private void initMap(){**

**Log.*d*(*TAG*, "initMap: initializing map");**

**SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.*map*);**

**mapFragment.getMapAsync(MapsActivity.this);**

**}**

**04. setRoute method**

***public void*** *setRoute()*

Description: set Route to Meeting location

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

**private void setRoute(){**

**Routing routing = new Routing.Builder()**

**.travelMode(Routing.TravelMode.*DRIVING*)**

**.withListener(this)**

**.waypoints(new LatLng(mLastKnownLocation.getLatitude(),mLastKnownLocation.getLongitude()),mLocationMeeting)**

**.build();**

**routing.execute();**

**}**

**05. onMarkerClick method**

***public void*** *onMarkerClick()*

Description: show marker location when click on marker button

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

**private void markerMeeting(LatLng latLng){**

**mLocationMeeting = latLng;**

**Log.*d*(*TAG*, "new click "+ latLng.toString());**

**List<Address> addresses = new ArrayList<>();**

**try {**

**if(mMarkerMeeting != null){**

**mMarkerMeeting.remove();**

**}**

**mMarkerMeeting = mMap.addMarker(new MarkerOptions().position(latLng));**

**mMarkerMeeting.setTag(latLng);**

**mMarkerMeeting.setTitle("meeting position");**

**addresses = geocoder.getFromLocation(latLng.latitude, latLng.longitude,1);**

**if(addresses.size()>0)**

**mMarkerMeeting.setSnippet("Address: " + addresses.get(0).getAddressLine(0) + "\n");**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

* 1. ***MessageActivity***

This class present for Message screen application.

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | TAG | String | "MessageActivity" | Private | the final variable |
| 02 | linaerlayoutChat | LinearLayout |  | Private | linearlayout of friends's list view |
| 03 | linearlayoutAddfriend | LinearLayout |  | Private | linearlayout of users's list view |
| 04 | tvAddFriend | TextView |  | Private | text view of add friend view |
| 05 | tvAddGroup | TextView |  | Private | text view of add group view |
| 06 | searchvSearch | SearchView |  | Private | search view |
| 07 | scrollViewMess | ScrollView |  | Private | scrollview for friend's list view |
| 08 | scrollViewAddFriend | ScrollView |  | Private | scrollview for user's list view |
| 09 | tvNameMessage | TextView |  | Private | text view of user's name |
| 10 | circleImageAvatar | CicleImageView |  | Private | User's avatar |
| 11 | mAuth | FirebaseAuth |  | Private | Authentication of Firebase |
| 12 | database | FirebaseDatabase |  | Private | Database of Firebase |
| 13 | id | int | 0 | Private | index of friend in list's friend |
| 14 | idAddUser | int | 0 | Private | index of user in list's user |

* **Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | onCreate | Initialize all activity of View in MessageActivity |
| 02 | onBackPressed | Override onBackPressed function |
| 03 | createChatView | Create a new view of friend in friend's list |
| 04 | createAddUserView | Create a new view of user in user's list |
| 05 | scaleView | Change size of view |
| 06 | AnimationforDeleteAndCancel | Set Animation for scale View |
| 07 | DisappearDeleteAndCancel | Set Animation disappear for scale View |
| 08 | SettingSearch | Set function for search view to find friend or user |
| 09 | SettingFriendChat | Set function for friend view to show up chat view, map view or delete it |
| 10 | SettingAddFriend | Set function for user view to send request or show up profile view |
| 11 | setUserNameToLayout | Set user's name on the top of view |
| 12 | LoadRequestedFriend | Connect to firebase to get list of users that be sent request |
| 13 | LoadFriendList | Connect to firebase to get list of friends |
| 14 | mapping | Set value to all variable view by findViewById function |
| 15 | setAvatar | Connect to firebase and load avatar of user |
| 16 | saveToInternalStorage | save Image downloaded to internal storage |

**01. onCreate method**

***protected void*** *onCreate(****final Bunbdle*** *savedInstanceState)*

Description: Initialize all activity of View in MessageActivity

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | savedInstanceState | Bundle | In |  | parameter to create activity |

* **Implementation**

**super**.onCreate(savedInstanceState);  
setContentView(R.layout.***activity\_message***);  
**this**.mapping();  
  
setUserNameToLayout();  
setAvatar(**mAuth**.getCurrentUser().getUid(), **circleImageAvatar**);  
  
ImageView imgvCallActivityProfile = findViewById(R.id.***image\_avatar***);  
imgvCallActivityProfile.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Intent intent = **new** Intent(MessageActivity.**this**, ProfileActivity.**class**);  
 startActivity(intent);  
 }  
});  
  
**id** = 0;  
**idAddUser** = 0;  
  
**tvAddFriend**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **scrollViewAddFriend**.setVisibility(View.***VISIBLE***);  
 **scrollViewMess**.setVisibility(View.***GONE***);  
 **searchvSearch**.setIconified(**false**);  
 }  
});  
  
**tvAddGroup**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **for** (**int** i = 0; i < **id**; i++) {  
 **final** Button cancel = findViewById(R.id.***cancel*** + i);  
 **final** Button delete = findViewById(R.id.***delete*** + i);  
 DisappearDeleteAndCancel(cancel, 0, 1);  
 DisappearDeleteAndCancel(delete, 1, 0);  
 }  
 CreateGroupDialog dialog = **new** CreateGroupDialog();  
 dialog.show(getSupportFragmentManager(), **null**);  
 }  
});  
  
SettingSearch();  
LoadFriendList();  
LoadRequestedFriend();

**02. onBackPressed method**

***public void*** *onBackPressed()*

Description: Override onBackPressed function

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

**if** (**scrollViewAddFriend**.getVisibility() == View.***VISIBLE***) {  
 **scrollViewAddFriend**.setVisibility(View.***GONE***);  
 **scrollViewMess**.setVisibility(View.***VISIBLE***);  
 **searchvSearch**.setQuery(**""**, **false**);  
 **searchvSearch**.setIconified(**true**);  
  
} **else if** (!**searchvSearch**.isIconified()) {  
 **searchvSearch**.setIconified(**true**);  
}  
**for** (**int** i = 0; i < **id**; i++) {  
 **final** Button cancel = findViewById(R.id.***cancel*** + i);  
 **final** Button delete = findViewById(R.id.***delete*** + i);  
 DisappearDeleteAndCancel(cancel, 0, 1);  
 DisappearDeleteAndCancel(delete, 1, 0);  
}

**03. createChatView method**

***public void*** *createChatView(****String*** *userName,* ***String*** *ID\_User,* ***String*** *NameOfUser,* ***String*** *type)*

Description: Create a new view of friend in friend's list

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | userName | String | In |  | friend's username or group's ID |
| 02 | ID\_User | String | In |  | friend's ID key on firebase or group's ID |
| 03 | NameOfUser | String | In |  | friend's full name or group's name |
| 04 | type | String | In |  | is Group or Friend chat |

* **Implementation**

RelativeLayout relativeLayout = **new** RelativeLayout(**this**);  
relativeLayout.setTag(userName);  
relativeLayout.setId(R.id.***ChatFriend*** + **id**);  
  
*// khởi tạo parameter cho relativeLayout (chat)*RelativeLayout.LayoutParams textparams = **new** RelativeLayout.LayoutParams(  
 RelativeLayout.LayoutParams.***MATCH\_PARENT***, RelativeLayout.LayoutParams.***WRAP\_CONTENT***);  
textparams.**topMargin** = 20;  
textparams.**rightMargin** = 20;  
textparams.**leftMargin** = 20;  
*// tạo ra TextView*TextView textviewUserName = **new** TextView(**this**);  
textviewUserName.setBackground(getDrawable(R.drawable.***input\_text\_chat***));  
textviewUserName.setId(**id**);  
textviewUserName.setTag(type);  
textviewUserName.setHeight(150);  
textviewUserName.setGravity(Gravity.***CENTER***);  
textviewUserName.setTextSize(20);  
textviewUserName.setText(NameOfUser);  
*// Cho TextView vào parameter vừa tạo ra*relativeLayout.addView(textviewUserName, textparams);  
  
*// khởi tạo parameter cho relativeLayout (avatar)*RelativeLayout.LayoutParams imageparams = **new** RelativeLayout.LayoutParams(  
 RelativeLayout.LayoutParams.***WRAP\_CONTENT***, RelativeLayout.LayoutParams.***WRAP\_CONTENT***);  
imageparams.**topMargin** = 20;  
imageparams.**leftMargin** = 20;  
imageparams.addRule(RelativeLayout.***ALIGN\_PARENT\_LEFT***);  
*// tạo ra ImageView*de.hdodenhof.circleimageview.CircleImageView image = **new** de.hdodenhof.circleimageview.CircleImageView(MessageActivity.**this**);  
relativeLayout.addView(image, imageparams);  
**if** (type.equals(**"Group Chat"**)) image.setImageResource(R.drawable.***ic\_request\_friend***);  
**else** setAvatar(ID\_User, image);  
image.setTag(textviewUserName.getText() + **"Avatar"**);  
image.getLayoutParams().**height** = 150;  
image.getLayoutParams().**width** = 150;  
image.setBorderColor(Color.*parseColor*(**"#FFFFFFFF"**));  
image.setBorderWidth(3);  
*// Cho ImageView vào parameter vừa tạo ra  
  
// khởi tạo parameter cho relativeLayout (location)*RelativeLayout.LayoutParams locationparams = **new** RelativeLayout.LayoutParams(  
 RelativeLayout.LayoutParams.***WRAP\_CONTENT***, RelativeLayout.LayoutParams.***WRAP\_CONTENT***);  
locationparams.**topMargin** = 50;  
locationparams.**rightMargin** = 30;  
locationparams.addRule(RelativeLayout.***ALIGN\_PARENT\_RIGHT***);  
*// tạo ra TextView*TextView textviewLocation = **new** TextView(**this**);  
textviewLocation.setId(R.id.***location*** + **id**);  
textviewLocation.setTag(textviewUserName.getText() + **"Location"**);  
textviewLocation.setBackgroundResource(R.drawable.***ic\_locate***);  
textviewLocation.setHeight(100);  
textviewLocation.setWidth(100);  
*// Cho ImageView vào parameter vừa tạo ra*relativeLayout.addView(textviewLocation, locationparams);  
  
*// khởi tạo parameter cho relativeLayout (Delete)*RelativeLayout.LayoutParams deleteparams = **new** RelativeLayout.LayoutParams(  
 250, 120);  
deleteparams.**topMargin** = 40;  
deleteparams.**rightMargin** = 150;  
deleteparams.addRule(RelativeLayout.***ALIGN\_PARENT\_RIGHT***);  
*// tạo ra TextView*Button btnDelete = **new** Button(**this**);  
btnDelete.setTag(textviewUserName.getText() + **"Delete"**);  
btnDelete.setId(R.id.***delete*** + **id**);  
btnDelete.setText(**"Delete"**);  
btnDelete.setGravity(Gravity.***CENTER***);  
btnDelete.setTextSize(15);  
btnDelete.setBackgroundResource(R.drawable.***button\_v3***);  
btnDelete.setVisibility(View.***GONE***);  
*// Cho ImageView vào parameter vừa tạo ra*relativeLayout.addView(btnDelete, deleteparams);  
  
*// khởi tạo parameter cho relativeLayout (Cancel)*RelativeLayout.LayoutParams cancelparams = **new** RelativeLayout.LayoutParams(  
 250, 120);  
cancelparams.**topMargin** = 40;  
cancelparams.**leftMargin** = 170;  
  
*// tạo ra TextView*Button btnCancel = **new** Button(**this**);  
btnCancel.setTag(textviewUserName.getText() + **"Cancel"**);  
btnCancel.setId(R.id.***cancel*** + **id**);  
btnCancel.setText(**"Cancel"**);  
btnCancel.setGravity(Gravity.***CENTER***);  
btnCancel.setTextSize(15);  
btnCancel.setBackgroundResource(R.drawable.***button\_v2***);  
btnCancel.setVisibility(View.***GONE***);  
*// Cho ImageView vào parameter vừa tạo ra*relativeLayout.addView(btnCancel, cancelparams);  
  
*// thêm thanh chat mới vào message***linearlayoutChat**.addView(relativeLayout);  
**id**++;

**04. createAddUserView method**

***public******void*** *createAddUserView(****String*** *userName,* ***String*** *ID\_User,* ***String*** *NameOfUser)*

Description: Create a new view of user in user's list

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | userName | String | In |  | user's username |
| 02 | ID\_User | String | In |  | user's ID key on firebase |
| 03 | NameOfUser | String | In |  | user's full name |

* **Implementation**

RelativeLayout relativeLayout = **new** RelativeLayout(**this**);  
relativeLayout.setTag(userName);  
relativeLayout.setId(R.id.***RelativeLAddFriend*** + **idAddUser**);  
relativeLayout.setVisibility(View.***GONE***);  
  
*// khởi tạo parameter cho relativeLayout (user)*RelativeLayout.LayoutParams textparams = **new** RelativeLayout.LayoutParams(  
 RelativeLayout.LayoutParams.***MATCH\_PARENT***, RelativeLayout.LayoutParams.***WRAP\_CONTENT***);  
textparams.**topMargin** = 20;  
textparams.**rightMargin** = 20;  
textparams.**leftMargin** = 20;  
*// tạo ra TextView*TextView textviewUserName = **new** TextView(**this**);  
textviewUserName.setBackgroundColor(Color.*parseColor*(**"#FFFFFF"**));  
textviewUserName.setBackground(getDrawable(R.drawable.***line***));  
textviewUserName.setHeight(150);  
textviewUserName.setGravity(Gravity.***CENTER***);  
textviewUserName.setTextSize(20);  
textviewUserName.setText(NameOfUser);  
textviewUserName.setId(R.id.***txtAddFriend*** + **idAddUser**);  
*// Cho TextView vào parameter vừa tạo ra*relativeLayout.addView(textviewUserName, textparams);  
  
*// khởi tạo parameter cho relativeLayout (avatar)*RelativeLayout.LayoutParams imageparams = **new** RelativeLayout.LayoutParams(  
 RelativeLayout.LayoutParams.***WRAP\_CONTENT***, RelativeLayout.LayoutParams.***WRAP\_CONTENT***);  
imageparams.**topMargin** = 20;  
imageparams.**leftMargin** = 20;  
imageparams.addRule(RelativeLayout.***ALIGN\_PARENT\_LEFT***);  
*// tạo ra ImageView*de.hdodenhof.circleimageview.CircleImageView image = **new** de.hdodenhof.circleimageview.CircleImageView(MessageActivity.**this**);  
relativeLayout.addView(image, imageparams);  
setAvatar(ID\_User, image);  
image.setTag(textviewUserName.getText() + **"Avatar"**);  
image.setBorderWidth(3);  
image.setBorderColor(Color.*parseColor*(**"#FFB3B2B2"**));  
image.getLayoutParams().**height** = 150;  
image.getLayoutParams().**width** = 150;  
*// Cho ImageView vào parameter vừa tạo ra  
  
// khởi tạo parameter cho relativeLayout (add)*RelativeLayout.LayoutParams addparams = **new** RelativeLayout.LayoutParams(  
 210, 120);  
addparams.**topMargin** = 40;  
addparams.**rightMargin** = 30;  
addparams.addRule(RelativeLayout.***ALIGN\_PARENT\_RIGHT***);  
*// tạo ra TextView*Button btnAdd = **new** Button(**this**);  
btnAdd.setId(R.id.***addButton*** + **idAddUser**);  
btnAdd.setGravity(Gravity.***CENTER***);  
btnAdd.setPaddingRelative(10, 10, 0, 0);  
btnAdd.setTag(textviewUserName.getText() + **"Add"**);  
btnAdd.setText(**"Add"**);  
btnAdd.setBackgroundResource(R.drawable.***button\_v2***);  
btnAdd.setTextSize(15);  
*// Cho ImageView vào parameter vừa tạo ra*relativeLayout.addView(btnAdd, addparams);  
  
*// thêm thanh chat mới vào message***linearlayoutAddfriend**.addView(relativeLayout);  
**idAddUser**++;

**05. scaleView method**

***public void*** *scaleView(****View*** *v,* ***float*** *startScaleX,* ***float*** *endScaleX,* ***float*** *pivotX,* ***float*** *pivot Y)*

Description: Change size of view

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | v | View | In |  | view to scale |
| 02 | startScaleX | float | In |  | start vaulues for the X axis scaling |
| 03 | endScaleX | float | In |  | end values for the Y axis scaling |
| 04 | pivotX | float | In |  | pivot point of X scaling |
| 05 | pivotY | float | In |  | Pivot point of Y scaling |

* **Implementation**

Animation anim = **new** ScaleAnimation(  
 startScaleX, endScaleX, *// Start and end values for the X axis scaling* (**float**) 1, (**float**) 1, *// Start and end values for the Y axis scaling* Animation.***RELATIVE\_TO\_SELF***, pivotX, *// Pivot point of X scaling* Animation.***RELATIVE\_TO\_SELF***, pivotY); *// Pivot point of Y scaling*anim.setFillAfter(**false**); *// Needed to keep the result of the animation*anim.setDuration(300);  
v.startAnimation(anim);

**06. AnimationforDeleteAndCancel method**

***public void*** *AnimationforDeleteAndCancel(****View*** *v,* ***float*** *pivotX,* ***float*** *pivotY)*

Description: Set Animation for scale View

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | v | View | In |  | view to scale |
| 04 | pivotX | float | In |  | pivot point of X scaling |
| 05 | pivotY | float | In |  | Pivot point of Y scaling |

* **Implementation**

**if** (v.getVisibility() == View.***GONE***) {  
 v.setVisibility(View.***VISIBLE***);  
 scaleView(v, 0, 1, pivotX, pivotY);  
} **else if** (v.getVisibility() == View.***VISIBLE***) {  
 v.setVisibility(View.***GONE***);  
 scaleView(v, 1, 0, pivotX, pivotY);  
}

**07. DisappearDeleteAndCancel method**

***public void*** *DisappearDeleteAndCancel(****View*** *v,* ***float*** *pivotX,* ***float*** *pivotY)*

Description: Set Animation disappear for scale View

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | v | View | In |  | view to scale |
| 02 | pivotX | float | In |  | pivot point of X scaling |
| 03 | pivotY | float | In |  | Pivot point of Y scaling |

* **Implementation**

**if** (v.getVisibility() == View.***VISIBLE***) {  
 v.setVisibility(View.***GONE***);  
 scaleView(v, 1, 0, pivotX, pivotY);  
}

**08. SettingSearch method**

***public void*** *SettingSearch()*

Description: Set function for search view to find friend or user

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

*// Search friend's list***searchvSearch**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **searchvSearch**.setIconified(**false**);  
 **for** (**int** i = 0; i < **id**; i++) {  
 **final** Button cancel = findViewById(R.id.***cancel*** + i);  
 **final** Button delete = findViewById(R.id.***delete*** + i);  
 DisappearDeleteAndCancel(cancel, 0, 1);  
 DisappearDeleteAndCancel(delete, 1, 0);  
 }  
 }  
});  
*// Search friend's list***searchvSearch**.setOnQueryTextListener(**new** android.support.v7.widget.SearchView.OnQueryTextListener() {  
 @Override  
 **public boolean** onQueryTextSubmit(String query) {  
 **searchvSearch**.clearFocus();  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onQueryTextChange(String newText) {  
 **if** (!newText.isEmpty()) {  
 callSearch(newText);  
 } **else** {  
 **if** (**scrollViewMess**.getVisibility() == View.***VISIBLE***) {  
 **for** (**int** i = 0; i < **id**; i++) {  
 RelativeLayout relativeLayout = findViewById(R.id.***ChatFriend*** + i);  
 **if** (relativeLayout.getTag() != **"@Delete"**)  
 relativeLayout.setVisibility(View.***VISIBLE***);  
 }  
 } **else** { *// scrollViewAddfriend visible* **for** (**int** i = 0; i < **idAddUser**; i++) {  
 RelativeLayout relativeLayout = findViewById(R.id.***RelativeLAddFriend*** + i);  
 relativeLayout.setVisibility(View.***VISIBLE***);  
 }  
 }  
 }  
 **return true**;  
 }  
  
 **void** callSearch(**final** String query) {  
 *//Do searching* showProgressDialog();  
 **if** (**scrollViewMess**.getVisibility() == View.***VISIBLE***) {  
 **for** (**int** i = 0; i < **id**; i++) {  
 RelativeLayout relativeLayout = findViewById(R.id.***ChatFriend*** + i);  
 TextView textViewUsername = findViewById(i);  
 **if** (!textViewUsername.getText().toString().contains(query) && !relativeLayout.getTag().toString().contains(query))  
 relativeLayout.setVisibility(View.***GONE***);  
 **else** relativeLayout.setVisibility(View.***VISIBLE***);  
 }  
 hideProgressDialog();  
 } **else** { *// scrollViewAddfriend visible  
 /\*\*  
 \* get User by query in database and createAddUserView  
 \* \*/* **final** FirebaseAuth mAuth = FirebaseAuth.*getInstance*();  
  
 **final** String username = mAuth.getCurrentUser().getDisplayName();  
 DatabaseReference ref = **database**.getReference(**"User"**);  
  
 ref.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot snapshot) {  
 **for** (DataSnapshot db : snapshot.getChildren()) {  
 **final** User user = db.getValue(User.**class**);  
 **final** String ID\_user = db.getKey();  
 **if** (user != **null** && !user.getUserName().equals(username)) {  
 **if** (user.getUserName().contains(query) ||

user.getNameOfUser().contains(query)) {  
 **boolean** isExist = **false**;  
 **for** (**int** i = 0; i < **idAddUser**; i++) {  
 RelativeLayout relativeLayout =

findViewById(R.id.***RelativeLAddFriend*** + i);  
 **if**(user.getUserName().equals(relativeLayout.getTag().

toString())) {  
 isExist = **true**;  
 **break**;  
 }  
 }  
 **if** (!isExist) {  
 createAddUserView(user.getUserName(), ID\_user,

user.getNameOfUser());  
 SettingAddFriend();  
 **for** (**int** i = 0; i < **id**; i++) {  
 RelativeLayout relativeLayout =

findViewById(R.id.***ChatFriend*** + i);  
 **if**(!relativeLayout.getTag().toString()

.equals(**"@Delete"**)) {  
 **if** (relativeLayout.getTag().toString()

.equals(user.getUserName())) {  
 **final** Button btnAdd =

findViewById(R.id.***addButton*** + **idAddUser** -

1);  
 btnAdd.setText(**"Friend"**);  
 btnAdd.setBackgroundResource(

R.drawable.***button\_v2***);  
 **break**;  
 }  
 }  
 }  
 }  
 DatabaseReference refRequestFriend =

**database**.getReference(**"requestFriendList"**);  
 refRequestFriend.child(mAuth.getCurrentUser().getUid())  
 .addListenerForSingleValueEvent(**new**

ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot

dataSnapshot) {  
  
 **if** (!dataSnapshot.exists()) **return**;  
 **if** (dataSnapshot.hasChild(ID\_user)) {  
 String userName\_User =

dataSnapshot.child(ID\_user).

getValue(String.**class**);  
 **for** (**int** i = 0; i < **idAddUser**; i++) {  
 RelativeLayout relativeLayout =

findViewById(R.id.***RelativeLAddFriend*** + i);  
 **if** (relativeLayout.getTag().toString()

.equals(userName\_User)) {  
 **final** Button btnAdd =

findViewById(R.id.***addButton*** + i);  
 btnAdd.setText(**"Request"**);  
 btnAdd.setBackgroundResource(

R.drawable.***button***);  
 **break**;  
 }  
 }  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
 });  
 }  
 }  
 }  
 **for** (**int** i = 0; i < **idAddUser**; i++) {  
 RelativeLayout relativeLayout = findViewById(R.id.***RelativeLAddFriend*** + i);  
 TextView textViewAddFriend = findViewById(R.id.***txtAddFriend*** + i);  
 **if** (!relativeLayout.getTag().toString().contains(query) &&

!textViewAddFriend.getText().toString().contains(query))  
 relativeLayout.setVisibility(View.***GONE***);  
 **else** relativeLayout.setVisibility(View.***VISIBLE***);  
 }  
 hideProgressDialog();  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 hideProgressDialog();  
 }  
 });  
 }  
 }  
});

**09. SettingFriendChat method**

***public void*** *SettingFriendChat()*

Description: Set function for friend view to show up chat view, map view or delete it

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

*// chat view***final** String ID\_User = **mAuth**.getCurrentUser().getUid();  
**int** i = **id** - 1;  
**final** RelativeLayout relativeLayout = findViewById(R.id.***ChatFriend*** + i);  
**final** Button cancel = findViewById(R.id.***cancel*** + i);  
**final** Button delete = findViewById(R.id.***delete*** + i);  
**final** TextView location = findViewById(R.id.***location*** + i);  
**final** TextView chatView = findViewById(i);  
  
relativeLayout.setOnLongClickListener(**new** View.OnLongClickListener() {  
 @Override  
 **public boolean** onLongClick(View v) {  
 AnimationforDeleteAndCancel(cancel, 0, 1);  
 AnimationforDeleteAndCancel(delete, 1, 0);  
 **return true**;  
 }  
 });  
  
 chatView.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **final** Intent intent = **new** Intent(MessageActivity.**this**, ChatActivity.**class**);  
 **if** (chatView.getTag().toString().equals(**"Friend Chat"**)) {  
 DatabaseReference refUserKey = **database**.getReference(**"userKey"**).

child(relativeLayout.getTag().toString());  
 refUserKey.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **final** String ID\_Friend = dataSnapshot.getValue(String.**class**);  
 *//------------------------------------------------------------------  
 // Truyền dữ liệu cho ChatActivity Friend chat* intent.putExtra(**"visit\_user\_id"**, ID\_Friend);  
 intent.putExtra(**"user\_name"**, chatView.getText().toString());  
 intent.putExtra(**"type\_chat"**, chatView.getTag().toString());  
 startActivity(intent);  
 *//------------------------------------------------------------------* }  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }});  
 } **else** {  
 *//----------------------------------------------------------------------  
 // Truyền dữ liệu cho ChatActivity Group chat* intent.putExtra(**"visit\_user\_id"**, relativeLayout.getTag().toString());  
 intent.putExtra(**"user\_name"**, chatView.getText().toString());  
 intent.putExtra(**"type\_chat"**, chatView.getTag().toString());  
 startActivity(intent);  
 *//----------------------------------------------------------------------* }  
 }  
 });  
 chatView.setOnLongClickListener(**new** View.OnLongClickListener() {  
 @Override  
 **public boolean** onLongClick(View view) {  
 AnimationforDeleteAndCancel(cancel, 0, 1);  
 AnimationforDeleteAndCancel(delete, 1, 1);  
 **return true**;  
 }  
 });  
 cancel.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 scaleView(cancel, 1, 0, 0, 1);  
 scaleView(delete, 1, 0, 1, 1);  
 view.setVisibility(View.***GONE***);  
 delete.setVisibility(View.***GONE***);  
 }  
 });  
 delete.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 Database db = **new** Database();  
 **if** (chatView.getTag().toString().equals(**"Friend Chat"**))  
 db.deleteFriend(relativeLayout.getTag().toString());  
 **else** db.deleteGroup(relativeLayout.getTag().toString());  
  
 relativeLayout.setVisibility(View.***GONE***);  
 relativeLayout.setTag(**"@Delete"**);  
 }  
 });  
 location.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **final** Intent intent = **new** Intent(MessageActivity.**this**,

MapsActivity.**class**);  
 *// is Friend Chat* **if** (chatView.getTag().toString().equals(**"Friend Chat"**)) {  
 DatabaseReference refUserKey = **database**.getReference(**"userKey"**).

child(relativeLayout.getTag().toString());  
 refUserKey.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **final** String ID\_Friend =dataSnapshot.getValue(String.**class**);  
 **final** DatabaseReference refMeeting = **database**.getReference(

**"meeting"**).child(ID\_User + ID\_Friend);  
 refMeeting.addListenerForSingleValueEvent(**new**

ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **if** (dataSnapshot.hasChild(**"latitude"**) &&

dataSnapshot.hasChild(**"longitude"**)) {  
 intent.putExtra(**"mConverId"**, ID\_User +

ID\_Friend);  
 startActivity(intent);  
 } **else** {  
 Toast.*makeText*(MessageActivity.**this**, **"Do not**

**have the meeting"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 @Override  
 **public void** onCancelled(DatabaseError databaseError){});  
 }  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {}  
 });  
 }  
 *// is Group Chat* **else** {  
 **final** DatabaseReference refMeeting = **database**.getReference(

**"meeting"**).child(relativeLayout.getTag().toString());  
 refMeeting.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {**if** (dataSnapshot.hasChild(**"latitude"**) && dataSnapshot.

hasChild(**"longitude"**)) {  
 intent.putExtra(**"mConverId"**,

relativeLayout.getTag().toString());  
 startActivity(intent);  
 } **else** {  
 Toast.*makeText*(MessageActivity.**this**, **"Do not have the**

**meeting"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
 });  
 }  
 }  
  
 });  
  
 location.setOnLongClickListener(**new** View.OnLongClickListener() {  
 @Override  
 **public boolean** onLongClick(View view) {  
 AnimationforDeleteAndCancel(cancel, 0, 1);  
 AnimationforDeleteAndCancel(delete, 1, 1);  
 **return true**;  
 }  
 });

**10. SettingAddFriend method**

*public void DisappearDeleteAndCancel(View v, float pivotX, float pivotY)*

Description: Set Animation disappear for scale View

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

Log.*d*(***TAG***, **"SETTTINGFRIEND: BEGIN"**);  
*// chat view***int** i = **idAddUser** - 1;  
**final** RelativeLayout relativeLayout = findViewById(R.id.***RelativeLAddFriend*** + i);  
**final** TextView textViewAddFriend = findViewById(R.id.***txtAddFriend*** + i);  
**final** Button btnAdd = findViewById(R.id.***addButton*** + i);  
  
btnAdd.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **if** (btnAdd.getText().equals(**"Add"**)) {  
 Database mydb = **new** Database();  
 Log.*d*(***TAG***, **"SENDREQUESTFRIEND: IN"**);  
 mydb.sendRequestFriend(relativeLayout.getTag().toString());  
  
 btnAdd.setText(**"Sent"**);  
 btnAdd.setBackgroundResource(R.drawable.***button\_v4***);  
 } **else if** (btnAdd.getText().equals(**"Sent"**)) {  
 Database mydb = **new** Database();  
 mydb.cancelRequestFriend(relativeLayout.getTag().toString());  
 btnAdd.setText(**"Add"**);  
 btnAdd.setBackgroundResource(R.drawable.***button\_v2***);  
 }  
 }  
});  
  
textViewAddFriend.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **final** Intent intent = **new** Intent(MessageActivity.**this**, ViewProfileActivity.**class**);  
 **final** String userName\_other = relativeLayout.getTag().toString();  
 DatabaseReference refUserKey = **database**.getReference(**"userKey"**).child(userName\_other);  
 refUserKey.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 String ID\_other = dataSnapshot.getValue(String.**class**);  
 intent.putExtra(**"userID"**, ID\_other);  
 startActivity(intent);  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
 });  
 }  
});

**11. setUserNameToLayout method**

***public void*** *setUserNameToLayout()*

Description: Set user's name on the top of view

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

**database**.getReference(**"User"**).child(**mAuth**.getCurrentUser().getUid()).child(**"nameOfUser"**).addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **tvNameMessage**.setText(dataSnapshot.getValue(String.**class**));  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
});

**12. LoadRequestedFriend method**

***public void*** *LoadRequestedFriend()*

Description: Connect to firebase to get list of users that be sent request

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

showProgressDialog();  
String id\_user = **mAuth**.getCurrentUser().getUid();  
DatabaseReference ref = **database**.getReference(**"senderRequestFriend"**).child(id\_user);  
  
ref.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot snapshot) {  
 **for** (DataSnapshot db : snapshot.getChildren()) {  
 DatabaseReference refUser = **database**.getReference(**"User"**).child(db.getKey());  
 **final** String ID\_Other = db.getKey();  
 refUser.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 User otherUser = dataSnapshot.getValue(User.**class**);  
 createAddUserView(otherUser.getUserName(), ID\_Other, otherUser.getNameOfUser());  
 Button btnAdd = findViewById(R.id.***addButton*** + **idAddUser** - 1);  
 btnAdd.setText(**"Sent"**);  
 btnAdd.setBackgroundResource(R.drawable.***button\_v4***);  
 SettingAddFriend();  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
 });  
 }  
 hideProgressDialog();  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 hideProgressDialog();  
 }  
});

**13. LoadFriendList method**

***public void*** *LoadFriendList()*

Description: Connect to firebase to get list of friends

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

*// LoadFriend create friend chat*showProgressDialog();  
String Username = **mAuth**.getCurrentUser().getDisplayName();  
**final** String ID\_User = **mAuth**.getCurrentUser().getUid();  
**if** (Username == **null**) **return**;  
DatabaseReference ref = **database**.getReference(**"Chat"**).child(ID\_User);  
ref.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **for** (DataSnapshot db : dataSnapshot.getChildren()) {  
 **final** String ID\_Other = db.getKey();  
 **final** Conversation conversation = db.getValue(Conversation.**class**);  
 **final** String UserName\_Other = conversation.getConversationName();  
  
 **if** (conversation.getType().equals(**"Friend Chat"**)) {  
 DatabaseReference refUser = **database**.getReference(**"User"**).child(ID\_Other);  
 refUser.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 User FriendInfo = dataSnapshot.getValue(User.**class**);  
 createChatView(UserName\_Other, ID\_Other, FriendInfo.getNameOfUser(), conversation.getType());  
 SettingFriendChat();  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
 });  
 } **else** {  
 *// CreateChatView Group conversation* createChatView(db.getKey(), db.getKey(), UserName\_Other, conversation.getType());  
 SettingFriendChat();  
 }  
 }  
 hideProgressDialog();  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 hideProgressDialog();  
 }  
});

**14. mapping method**

***private void*** *mapping()*

Description: Set value to all variable view by findViewById function

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 |  |  |  |  |  |

* **Implementation**

**tvAddFriend** = findViewById(R.id.***textview\_add\_friend\_message***);  
**tvAddGroup** = findViewById(R.id.***textview\_add\_group\_message***);  
**searchvSearch** = findViewById(R.id.***searchview\_Message***);  
**tvNameMessage** = findViewById(R.id.***textview\_NameMessage***);  
**circleImageAvatar** = findViewById(R.id.***image\_avatar***);  
**scrollViewAddFriend** = findViewById(R.id.***scrollviewAddfriend\_Message***);  
**scrollViewMess** = findViewById(R.id.***scrollView\_Message***);  
**linearlayoutChat** = findViewById(R.id.***Linearlayoutchat\_Message***);  
**linearlayoutAddfriend** = findViewById(R.id.***LinearlayoutAddfriend\_Message***);

**15. setAvatar method**

***private void*** *setAvatar(final String UserID, final CircleImageView image)*

Description: Connect to firebase and load avatar of user

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | *UserID* | *String* | In |  | ID of user |
| 02 | *image* | *CircleImageView* | In |  | avatar of user |

* **Implementation**

showProgressDialog();  
**final** String AVATAR = **"avatar"**;  
File localFile = **null**;  
**try** {  
 localFile = File.*createTempFile*(**"images"**, **"jpg"**);  
} **catch** (IOException e) {  
 e.printStackTrace();  
}  
**final** File finalLocalFile = localFile;  
**if** (localFile != **null**) {  
 FirebaseStorage.*getInstance*().getReference()  
 .child(AVATAR + **"/"** + UserID)  
 .getFile(localFile)  
 .addOnSuccessListener(**new** OnSuccessListener<FileDownloadTask.TaskSnapshot>() {  
 @Override  
 **public void** onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {  
 *// Successfully downloaded data to local file* image.setImageBitmap(BitmapFactory.*decodeFile*(finalLocalFile.getAbsolutePath()));  
 hideProgressDialog();  
 }  
 })  
 .addOnFailureListener(**new** OnFailureListener() {  
 @Override  
 **public void** onFailure(@NonNull Exception exception) {  
 *// Handle failed download* image.setImageResource(R.drawable.ic\_user);  
 hideProgressDialog();  
 }  
 });  
}

**16. saveToInternalStorage method**

***public void*** *saveToInternalStorage(****Bitmap*** *bitmap,* ***String*** *userId)*

Description: save Image downloaded to internal storage

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | *bitmap* | *Bitmap* | In |  | bitmap to handle image |
| 02 | *userId* | *String* | In |  | user' id on firebase |

* **Implementation**

ContextWrapper contextWrapper = **new** ContextWrapper(getApplicationContext());  
*// path to /data/data/your app/app\_data/avatar*File directory = contextWrapper.getDir(**"avatar"**, Context.***MODE\_PRIVATE***);  
  
*// Create imageDir*File mypath = **new** File(directory, userId + **".jpg"**);  
  
FileOutputStream fos = **null**;  
**try** {  
 fos = **new** FileOutputStream(mypath);  
 *// Use the compress method on the BitMap object to write image to the OutputStream* bitmap.compress(Bitmap.CompressFormat.***PNG***, 100, fos);  
} **catch** (Exception e) {  
 e.printStackTrace();  
} **finally** {  
 **try** {  
 **if** (fos != **null**) {  
 fos.close();  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
}

* 1. ***ProfileActivity***

This class contains all information of user. And from here, user can change information.

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | mAuth | FirebaseAuth |  |  | Authentication of current user |
| 02 | mGoogleSignInClient | GoogleSignInClient |  |  | The current google login session |

* **Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | mapping | Map variables. |
| 02 | setData | Download information of user from firebase and display on the screen. |
| 03 | signOut | Sign out |
| 04 | applyPhonenumbers | Change user’s phone number |
| 05 | applyBitmapAvatar | Change user’s avatar |
| 06 | applyPassword | Change user’s password |

**01. mapping**

*public void mapping ()*

This is a method mapping variables of activity. It’s called first.

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

**02. setData**

*public void setData()*

This is a method downloading user’s information and then assign to variables displaying on the activity.

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

*if(checkImageExist(mAuth.getUid())){*

*Bitmap bitmap = loadImageFromStorage(mAuth.getUid());*

*imgAvatar.setImageBitmap(bitmap);*

*}*

*else{*

*File localFile = null;*

*try {*

*localFile = File.createTempFile("images", "jpg");*

*} catch (IOException e) {*

*e.printStackTrace();*

*}*

*final File finalLocalFile = localFile;*

*FirebaseStorage.getInstance().getReference()*

*.child(AVATAR + "/" + mAuth.getCurrentUser().getUid())*

*.getFile(localFile)*

*.addOnSuccessListener(new OnSuccessListener<FileDownloadTask.TaskSnapshot>() {*

*@Override*

*public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {*

*// Successfully downloaded data to local file*

*imgAvatar.setImageBitmap(BitmapFactory.decodeFile(finalLocalFile.getAbsolutePath()));*

*saveToInternalStorage(BitmapFactory.decodeFile(finalLocalFile.getAbsolutePath()), mAuth.getUid());*

*}*

*})*

*.addOnFailureListener(new OnFailureListener() {*

*@Override*

*public void onFailure(@NonNull Exception exception) {*

*// Handle failed download*

*// ...*

*}*

*});*

*}*

*FirebaseDatabase.getInstance()*

*.getReference(USER)*

*.child(mAuth.getCurrentUser().getUid())*

*.addValueEventListener(new ValueEventListener() {*

*@Override*

*public void onDataChange(DataSnapshot dataSnapshot) {*

*String displayName = dataSnapshot.child(FIRST\_NAME).getValue(String.class) + " " + dataSnapshot.child(LAST\_NAME).getValue(String.class);*

*txtName.setText(displayName);*

*txtGender.setText(dataSnapshot.child(GENDER).getValue(String.class));*

*txtBirthday.setText(dataSnapshot.child(DATE\_OF\_BIRTH).getValue(String.class));*

*txtEmail.setText(dataSnapshot.child(EMAIL).getValue(String.class));*

*txtPhone.setText(dataSnapshot.child(PHONE).getValue(String.class));*

*txtAddress.setText(dataSnapshot.child(ADDRESS).getValue(String.class));*

*password = dataSnapshot.child(PASSWORD).getValue(String.class);*

*btnCallActivityBack.setText(dataSnapshot.child(NAME\_OF\_USER).getValue(String.class));*

*}*

*@Override*

*public void onCancelled(DatabaseError databaseError) {*

*}*

*});*

**03. signOut**

*public void signOut()*

Log out current account

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

*mAuth.signOut();*

*// Google sign out*

*if(GoogleSignIn.getLastSignedInAccount(this) != null) {*

*mGoogleSignInClient.signOut().addOnCompleteListener(this,*

*new OnCompleteListener<Void>() {*

*@Override*

*public void onComplete(@NonNull Task<Void> task) {*

*//*

*}*

*});*

*}*

**04. applyPhonenumbers**

*public void applyPhonenumbers(String strPhonenumber)*

This is a method changing phone number

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | message | string | in |  | Message’s content |

* **Implementation**

*txtPhone.setText(strPhonenumber);*

*FirebaseDatabase.getInstance()*

*.getReference(USER)*

*.child(mAuth.getCurrentUser().getUid())*

*.child(PHONE)*

*.setValue(strPhonenumber);*

**05. applyBitmapAvatar**

*public void applyBitmapAvatar(String message, String senderId, long time)*

This is a method changing avatar image of user

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | Intent | Intent | in |  | Intent is passed from avatar change dialog |

* **Implementation**

*Bitmap bitmap = intent.getParcelableExtra("data");*

*imgAvatar.setImageBitmap(bitmap);*

*StorageReference storageRef = FirebaseStorage.getInstance().getReference();*

*ByteArrayOutputStream baos = new ByteArrayOutputStream();*

*bitmap.compress(Bitmap.CompressFormat.JPEG, 1, baos); //Anh modify quality 100 downto 25*

*byte[] data = baos.toByteArray();*

*UploadTask uploadTask = storageRef*

*.child(AVATAR + "/" + mAuth.getCurrentUser().getUid())*

*.putBytes(data);*

*uploadTask.addOnFailureListener(new OnFailureListener() {*

*@Override*

*public void onFailure(@NonNull Exception exception) {*

*// Handle unsuccessful uploads*

*}*

*}).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {*

*@Override*

*public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {*

*}*

*});*

*saveToInternalStorage(bitmap, mAuth.getUid());*

*hasChange = true;*

**05. applyPassword**

*public void applyPassword(boolean isSuccessful)*

This is a method displaying successful or not

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | isSuccessful | boolean | in |  | Whether the changing password is successful or not |

* **Implementation**

*if(isSuccessful){*

*Toast.makeText(this, "Change password successful!!!", Toast.LENGTH\_SHORT).show();*

*}*

*else{*

*Toast.makeText(this, "Change password fail!!!", Toast.LENGTH\_SHORT).show();*

*}*

* 1. ***Sign\_inActivity***

This class present for a sign in activity.

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | textvSignUp | TextView |  | Private |  |
| 02 | edUserName | EditText |  | Private |  |
| 03 | edPassword | EditText |  | Private |  |
| 04 | mAuth | FirebaseAuth |  | Private |  |
| 05 | mGoogleSignInClient | GoogleSignInClient |  | Private |  |
| 06 | db | FirebaseData |  | Private |  |

* **Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | onCreate | Initialize |
| 02 | signInEmail | Get device Location and show in map |
| 03 | signInUsername | Initilize map |
| 04 | validateForm | Check validate form |

**01. onCreate method**

*public void onCreate()*

Description: Initialize Sign In activity

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |

* **Implementation**

@Override

**protected void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_sign\_in***);

**textvSignUp** = findViewById(R.id.***TextView\_SignUp***);

**edUserName** = findViewById(R.id.***editText\_SignInUsername***);

**edPassWord** = findViewById(R.id.***editText\_SignInPassword***);

findViewById(R.id.***button\_mail\_signIn***).setOnClickListener(**this**);

findViewById(R.id.***button\_google\_signIn***).setOnClickListener(**this**);

**mAuth** = FirebaseAuth.*getInstance*();

*// Configure Google Sign In*

GoogleSignInOptions gso = **new** GoogleSignInOptions.Builder(GoogleSignInOptions.***DEFAULT\_SIGN\_IN***)

.requestIdToken(getString(R.string.***default\_web\_client\_id***))

.requestEmail()

.build();

**mGoogleSignInClient** = GoogleSignIn.*getClient*(**this**, gso);

}

**02. signInEmail method**

*public void signInEmail()*

Description: sign in with email

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |

* **Implementation**

**public void** singInEmail(**final** String email, String password){

Log.*d*(***TAG***, **"signIn: "**+email);

showProgressDialog(); **mAuth**.signInWithEmailAndPassword(email,password).addOnCompleteListener(Sign\_inActivity.**this**, **new** OnCompleteListener<AuthResult>() {

@Override

**public void** onComplete(@NonNull Task<AuthResult> task) {

**if** (task.isSuccessful()) {

*// Sign in success, update UI with the signed-in user's information*

Log.*d*(***TAG***, **"signInWithEmail:success"**);

FirebaseUser user = **mAuth**.getCurrentUser();

updateUI(user);

} **else** {

*// If sign in fails, display a message to the user.*

Log.*w*(***TAG***, **"signInWithEmail:failure"**, task.getException());

Toast.*makeText*(Sign\_inActivity.**this**, **"Authentication failed."**,

Toast.***LENGTH\_SHORT***).show();

updateUI(**null**);

}

hideProgressDialog();

}

});

}

**03. signInUserName method**

*public void signInUserName()*

Description: sign in with username

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |

* **Implementation**

**public void** singInUsername(String username, **final** String password){

DatabaseReference mapUser = **db**.getReference(**"MapUsername:Mail"**);

Log.*d*(***TAG***, **"signInWithUsername: "**+username);

showProgressDialog();

*// If sign in email fail, continue with as username.*

mapUser.child(username).addListenerForSingleValueEvent(**new** ValueEventListener() {

@Override

**public void** onDataChange(DataSnapshot dataSnapshot) {

**if**(dataSnapshot != **null**){

*// MapUsername newMap = dataSnapshot.getValue(MapUsername.class);*

String email = dataSnapshot.getValue(String.**class**);

**if**(email != **null** ) {

Log.*d*(***TAG***, **"RETURN EMAIL: "** + email);

singInEmail(email, password);

*//String userId = newMap.getUserId();*

}

**else**{

Log.*w*(***TAG***, **"signInWithUser:failure"**);

Toast.*makeText*(Sign\_inActivity.**this**, **"Authentication failed."**,

Toast.***LENGTH\_SHORT***).show();

updateUI(**null**);

}

}

**else**{

Log.*w*(***TAG***, **"signInWithUser:failure"**);

Toast.*makeText*(Sign\_inActivity.**this**, **"Authentication failed."**,

Toast.***LENGTH\_SHORT***).show();

updateUI(**null**);

}

hideProgressDialog();

}

@Override

**public void** onCancelled(DatabaseError databaseError) {

}

});

}

**04. validateForm method**

*public boolean validateForm()*

Description: check validate register form

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | <return> | Boolean | out |  | Return result of check form |

* **Implementation**

**public boolean** validateForm(){

**boolean** valid = **true**;

**if** (isEmpty(**edUserName**)) {

**edUserName**.setError(**"Invalid username"**);

**edUserName**.requestFocus();

valid =**false**;

}**else**

**edUserName**.setError(**null**);

**if** (isEmpty(**edPassWord**)) {

**edPassWord**.setError(**"Invalid password"**);

**edPassWord**.requestFocus();

valid =**false**;

} **else**

**edPassWord**.setError(**null**);

**return** valid;

}

* 1. ***Sign\_upActivity***

This class contains sign up activity

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | btnSignUp | Button |  | Private |  |
| 02 | etUserName | EditText |  | Private |  |
| 03 | etFirstName | EditText |  | Private |  |
| 04 | etLastName | EditText |  | Private |  |
| 05 | etPassword | EditText |  | Private |  |
| 06 | etConfirmPass | EditText |  | Private |  |
| 07 | etEmail | EditText |  | Private |  |
| 08 | etPhone | EditText |  | Private |  |
| 09 | etAddress | EditText |  | Private |  |
| 10 | mDateSetListener | DatePickerDialog.OnDateSetListener |  | Private |  |
| 11 | rdMale | RadioButton |  | Private |  |
| 12 | rdFemale | RadioButton |  | Private |  |
| 13 | mAuth | FirebaseAuth |  | Private |  |
| 14 | db | FirebaseDatabase |  | Private |  |

* **Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | onCreate | Initialize |
| 02 | createAccount | Create Account |
| 03 | validateForm | Check validate form |
| 04 | addAccountRealTimeDB | Get data from form |
| 05 | saveInfoUser | Save data to firebase |
| 06 | saveUsernameEmail | Save key pair username: Email to firebase |

**01. onCreate method**

*public void onCreate()*

Description: Initialize Sign up activity

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |

* **Implementation**

@Override

**protected void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_sign\_up***);

**etFirstName** = findViewById(R.id.***EditText\_SignUpFirstName***);

**etLastName** = findViewById(R.id.***EditText\_SignUpLastName***);

**etUserName** = findViewById(R.id.***EditText\_SignUpUserName***);

**etPassWord** = findViewById(R.id.***EditText\_SignUpPassword***);

**rdMale** = findViewById(R.id.***radioMale\_SignUp***);

**etEmail** = findViewById(R.id.***EditText\_SignUpEmail***);

**etPhone** = findViewById(R.id.***EditText\_SignUpPhone***);

**etAddress** = findViewById(R.id.***EditText\_SignUpAddress***);

**etConfirmPass** = findViewById(R.id.***EditText\_SignUpConfirmPassword***);

**textvSignIn** = findViewById(R.id.***TextView\_SignIn***);

**mDisplayDate** =findViewById(R.id.***DatePicker\_SignUp***);

**btnSignUp** = findViewById(R.id.***button\_SignUp***);

**mIcPicker** = findViewById(R.id.***ic\_day\_of\_birth***);

*// db*

**mAuth** = FirebaseAuth.*getInstance*();

}

**02. createAccount**

*Public void createAccount ()*

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

***public void*** *createAccount(String email, String password){*

Log.*d*(***TAG***, **"createAccount"**+ email);

showProgressDialog();

**mAuth**.createUserWithEmailAndPassword(email,password)

.addOnCompleteListener(Sign\_upActivity.**this**, **new** OnCompleteListener<AuthResult>() {

@Override

**public void** onComplete(@NonNull Task<AuthResult> task) {

**if** (task.isSuccessful()){

Log.*d*(***TAG***, **"createWithEmail: Success"**);

addAccountRealTimeDB();

hideProgressDialog();

Toast.*makeText*(Sign\_upActivity.**this**, **"Authentication success."**, Toast.***LENGTH\_SHORT***).show();

finish(); *// Anh modified*

Intent intent = **new** Intent(Sign\_upActivity.**this**, Sign\_inActivity.**class**);

startActivity(intent);

}

**else**{

Log.*w*(***TAG***, **"createWithEmail: Failure"**, task.getException());

Toast.*makeText*(Sign\_upActivity.**this**, **"Authentication failed."**, Toast.***LENGTH\_SHORT***).show();

}

hideProgressDialog();

}

});

}

**03. addAccountRealtimeDB**

*Public void addAccountRealTimeDB ()*

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

***public void*** *addAccountRealTimeDB(){*

***final*** *String firstname =* ***etFirstName****.getText().toString();*

***final*** *String lastname =* ***etLastName****.getText().toString();*

***final*** *String username =* ***etUserName****.getText().toString();*

*// final String password = etPassWord.getText().toString();*

***final*** *String gender =* ***rdMale****.isChecked() ?* ***"Male"*** *:* ***"Female"****;*

***final*** *String email =* ***etEmail****.getText().toString();*

***final*** *String phone =* ***etPhone****.getText().toString();*

***final*** *String birthday =* ***mDisplayDate****.getText().toString();*

***final*** *String address =* ***etAddress****.getText().toString();*

*User user =* ***new*** *User(username, firstname, lastname, gender, email, phone, address, birthday);*

*UserProfileChangeRequest profileUpdate =* ***new*** *UserProfileChangeRequest.Builder().setDisplayName(****etUserName****.getText().toString()).build();*

***mAuth****.getCurrentUser().updateProfile(profileUpdate);*

*String newUserId = saveInfoUser(user);*

*saveUsernameEmail(username, email, newUserId);*

*}*

**04. validateForm**

*public Boolean validateForm ()*

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | <return> | Boolean | out |  |  |

* **Implementation**

***public boolean*** *validateForm(){*

***boolean*** *valid =****true****;*

***if*** *(isEmpty(****etFirstName****)) {*

***etFirstName****.setError(****"First Name can't be empty"****);*

***etFirstName****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

***etFirstName****.setError(****null****);*

*}*

***if*** *(isEmpty(****etLastName****)) {*

***etLastName****.setError(****"Last Name can't be empty"****);*

***etLastName****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

***etLastName****.setError(****null****);*

*}*

***if*** *(isEmpty(****etUserName****)) {*

***etUserName****.setError(****"Username can't be empty"****);*

***etUserName****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

***etUserName****.setError(****null****);*

*}*

***if****(****etPassWord****.getText().toString().length() < 6){*

***etPassWord****.setError(****"Your password must have at least 6 letters, numbers and symbols"****);*

***etPassWord****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

***etPassWord****.setError(****null****);*

*}*

***if*** *(isEmpty(****etPassWord****)) {*

***etPassWord****.setError(****"PassWord can't be empty"****);*

***etPassWord****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

***if****(!****etPassWord****.getText().toString().equals(****etConfirmPass****.getText().toString())){*

***etConfirmPass****.setError(****"Confirm password not match"****);*

***etConfirmPass****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

***etPassWord****.setError(****null****);*

*}*

*}*

***if****(****mDisplayDate****.getText().toString().equals(****""****)){*

***mDisplayDate****.setError(****"Select Date of birth"****);*

***mDisplayDate****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

*Calendar cal = Calendar.getInstance();*

*String date =* ***mDisplayDate****.getText().toString();*

*String[] temp = date.split(****"/"****);*

***if****(Integer.parseInt(temp[2]) > cal.get(Calendar.****YEAR****) -5){*

***mDisplayDate****.setError(****"You too young to register. At least 5 year old"****);*

***mDisplayDate****.requestFocus();*

*}*

***else***

***mDisplayDate****.setError(****null****);*

*}*

***if****(!****rdMale****.isSelected()&& !****rdFemale****.isSelected()){*

***rdMale****.setError(****"Please select a gender"****);*

***rdFemale****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

***rdMale****.setError(****null****);*

*}*

***if*** *(isEmpty(****etEmail****)) {*

***etEmail****.setError(****"Email can't be empty"****);*

***etEmail****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

***if*** *(!isEmailValid(****etEmail****.getText().toString())){*

***etEmail****.setError(****"Please enter a valid email address"****);*

***etEmail****.requestFocus();*

*valid =* ***false****;*

*}****else*** *{*

***etEmail****.setError(****null****);*

*}*

*}*

***return*** *valid;*

*}*

**05. saveInfoUser**

*public String saveInfoUser()*

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | new\_user | User | In |  |  |
| 02 | <return> | String | Out |  |  |

* **Implementation**

***public*** *String saveInfoUser(User new\_user) {*

*DatabaseReference newUserID =* ***db****.getReference(****"User"****).child(****mAuth****.getCurrentUser().getUid());;*

***db****.getReference(****"userKey"****).child(new\_user.getUserName()).setValue(****mAuth****.getCurrentUser().getUid()); // Anh modify*

*newUserID.setValue(new\_user,* ***new*** *DatabaseReference.CompletionListener() {*

*@Override*

***public void*** *onComplete(DatabaseError databaseError, DatabaseReference databaseReference) {*

***if****(databaseError!=* ***null****)*

*Log.d(****TAG****,* ***"Write INFOUSER: ERROR "****+databaseError.getMessage());*

***else***

***Log.d(TAG, "Write INFOUSER: Successs");***

***}***

***});***

***return newUserID.getKey();***

***}***

**06. saveUsernameEmail**

*public void saveUsernameEmail()*

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | username | String | In |  |  |
| 02 | email | String | In |  |  |
| 03 | newUserId | String | In |  |  |

* **Implementation**

***public void*** *saveUsernameEmail(String username, String email,String newUserId){*

*Log.d(****TAG****,* ***"Write MapUSEr: Begin"****);*

*DatabaseReference mapUser =* ***db****.getReference(****"MapUsername:Mail"****);*

*mapUser.child(username).setValue(email,* ***new*** *DatabaseReference.CompletionListener() {*

*@Override*

***public void*** *onComplete(DatabaseError databaseError, DatabaseReference databaseReference) {*

***if****(databaseError!=* ***null****)*

*Log.d(****TAG****,* ***"Write MapUSer: ERROR "****+databaseError.getMessage());*

***else***

*Log.d(****TAG****,* ***"Write MapUSer: Successs"****);*

*}*

*});*

*}*

* 1. ***ViewProfileActivity***

This class has functions to view profile when one user wants to view information of another user.

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | userId | String |  |  | Id of user we want to display |

* **Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | mapping | Map variables |
| 02 | setData | Download some basic information of user on firebase and set to display |

**01. mapping method**

*public void* mapping*()*

Description: to map variables

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

*imgAvatar = findViewById(R.id.ImageView\_Avatar\_ViewProfile);*

*tvName = findViewById(R.id.TextView\_Name\_ViewProfile);*

*tvGender = findViewById(R.id.TextView\_Gender\_ViewProfile);*

*tvBirthday = findViewById(R.id.TextView\_BirthdayProfile);*

*btnBack = findViewById(R.id.Button\_Back\_ViewProfile);*

*Bundle bundle = getIntent().getExtras();*

*userID = bundle.containsKey("userID")? bundle.getString("userID")*

**02. setData** **method**

*public void* setData*()*

Description: download data from firebase through userId and display them

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
|  |  |  |  |  |  |

* **Implementation**

*File localFile = null;*

*try {*

*localFile = File.createTempFile("images", "jpg");*

*} catch (IOException e) {*

*e.printStackTrace();*

*}*

*final File finalLocalFile = localFile;*

*FirebaseStorage.getInstance().getReference()*

*.child(AVATAR + "/" + userID)*

*.getFile(localFile)*

*.addOnSuccessListener(new OnSuccessListener<FileDownloadTask.TaskSnapshot>() {*

*@Override*

*public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {*

*// Successfully downloaded data to local file*

*imgAvatar.setImageBitmap(BitmapFactory.decodeFile(finalLocalFile.getAbsolutePath()));*

*}*

*})*

*.addOnFailureListener(new OnFailureListener() {*

*@Override*

*public void onFailure(@NonNull Exception exception) {*

*// Handle failed download*

*// ...*

*}*

*});*

*FirebaseDatabase.getInstance()*

*.getReference(USER)*

*.child(userID)*

*.addValueEventListener(new ValueEventListener() {*

*@Override*

*public void onDataChange(DataSnapshot dataSnapshot) {*

*tvName.setText(dataSnapshot.child(NAME\_OF\_USER).getValue(String.class));*

*tvGender.setText(dataSnapshot.child(GENDER).getValue(String.class));*

*tvBirthday.setText(dataSnapshot.child(DATE\_OF\_BIRTH).getValue(String.class));*

*btnBack.setText(dataSnapshot.child(NAME\_OF\_USER).getValue(String.class));*

*}*

*@Override*

*public void onCancelled(DatabaseError databaseError) {*

*}*

*});*

* 1. ***Message\_Adapter***

This class contains functions of Chat View

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | mAuth | FirebaseAuth |  |  | The authentication of user |

**Methods**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | fetchMessage | Read messages from database |
| 02 | fetchAppointment | Read detail of appointment from database |

**01. fetchMessage**

*Public void fetchMessage ()*

This is a method to read messages from DB

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | <return> |  |  |  |  |

* **Implementation**

*<How to implement the method, it can be in pseudo code or activity diagram or just words>*

**02. fetchAppointment**

*Public string fetchAppointment ()*

This is a method read detail of appointment from DB

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | <return> |  |  |  |  |

* **Implementation**

*<How to implement the method, it can be in pseudo code or activity diagram or just words>*

* 1. ***CreateGroupDialog***

This class present for Dialog of searching friend to pick members for group

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | btnConfirm | Button |  | Private | button confirm creating group |
| 02 | btnCancel | Button |  | Private | button cancel creating group |
| 03 | edtNameOf Group | EditText |  | Private | text for input name of group |
| 04 | tvMembers | TextView |  | Private | Show member in friend's list has been chosen |
| 05 | viewCreateGroup | View |  | public | view of dialog creating group |

* ***Method***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | onCreateDialog | Initialize all activity of View in CreateGroupDialog |
| 02 | mapping | Set value to all variable view by findViewById function |

**01. onCreateDialog method**

***public Dialog*** *onCreateDialog(****Bundle*** *savedInstanceState)*

Description: Initialize all activity of View in CreateGroupDialog

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| **01** | *savedInstanceState* | *Bundle* | In |  | parameter to create activity |

* **Implementation**

AlertDialog.Builder builder = **new** AlertDialog.Builder(getActivity());  
  
LayoutInflater inflater = getActivity().getLayoutInflater();  
*viewCreateGroup* = inflater.inflate(R.layout.***dialog\_create\_group***, **null**);  
mapping(*viewCreateGroup*);  
  
builder.setView(*viewCreateGroup*);  
  
**btnConfirm**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 String strNameOfGroup = **edtNameOfGroup**.getText().toString();  
 String strMemebers = **tvMembers**.getText().toString();  
 String[] userName\_Member = strMemebers.split(**" {3}"**);  
 **if** (strNameOfGroup.isEmpty() || strMemebers.isEmpty()) {  
 Toast.*makeText*(getContext(), **"The fields must not be blank"**, Toast.***LENGTH\_SHORT***).show();  
 } **else if** (strMemebers.length() > 0) {  
 Database db = **new** Database();  
 db.CreateGroup(userName\_Member, strNameOfGroup);  
 *//--------------------------------------------------------------------------  
 // Do add view group here* getActivity().finish();  
 getActivity().startActivity(getActivity().getIntent());  
 *//--------------------------------------------------------------------------* Toast.*makeText*(getContext(), **"Create group successfully"**, Toast.***LENGTH\_SHORT***).show();  
 dismiss();  
 } **else** {  
 Toast.*makeText*(getContext(), **"Not Successfully"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
});  
**btnCancel**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 dismiss();  
 }  
});  
  
**tvMembers**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 SearchMemberDialog dialog = **new** SearchMemberDialog();  
 dialog.show(getChildFragmentManager(), **null**);  
 }  
});  
  
**return** builder.create();

**02. mapping method**

***private void*** *mapping(****View*** *view)*

Description: Set value to all variable view by findViewById function

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | view | View | In |  | set all view to variable in group dialog |

* **Implementation**

**btnConfirm** = view.findViewById(R.id.***button\_ConfirmCreateGroup***);  
**btnCancel** = view.findViewById(R.id.***button\_CancelCreateGroup***);  
**edtNameOfGroup** = view.findViewById(R.id.***EditText\_NameOfGroup***);  
**tvMembers** = view.findViewById(R.id.***TextView\_Members***);

* 1. ***SearchMemberDialog***

This class contains functions of Login and Register View

* **Attributes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Attribute** | **Type** | **Default** | **Note** | **Description** |
| 01 | btnOK | Button |  | Private | confirm Member chosen |
| 02 | btnCancel | Button |  | Private | cancel choosing member |
| 03 | searchvSearchMember | SearchView |  | Private | search friend in list to pick member |
| 04 | linearlayoutMember | LinearLayout |  | Private | layout contains friend's list |
| 05 | BLANK\_SIZE | int | 3 | Private | the space between every member's name |
| 06 | id | int | 0 | Private | index of friend in list's friend |

* ***Method***

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| 01 | onCreateDialog | Initialize all activity of View in SearchMemberDialog |
| 02 | createMemberView | Create a new view of friend in friend's list |
| 03 | mapping | Set value to all variable view by findViewById function |
| 04 | setAvatar | Connect to firebase and load avatar of user |
| 05 | saveToInternalStorage | save Image downloaded to internal storage |
| 06 | checkImageExist | check image if it has been in the storage |
| 07 | loadImageFromStorage | load image from the storage |
| 08 | LoadFriend | connect to firebase to get list of friend |

**01. onCreateDialog method**

***public Dialog*** *onCreateDialog(****Bundle*** *savedInstanceState)*

Description: Initialize all activity of View in CreateGroupDialog

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| **01** | *savedInstanceState* | *Bundle* | In |  | parameter to create activity |

* **Implementation**

AlertDialog.Builder builder = **new** AlertDialog.Builder(getActivity());  
  
LayoutInflater inflater = getActivity().getLayoutInflater();  
**final** View view = inflater.inflate(R.layout.***dialog\_search\_member***, **null**);  
mapping(view);  
**id** = 0;  
builder.setView(view);  
  
**searchvSearchMember**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 **searchvSearchMember**.setIconified(**false**);  
 }  
});  
*// Search friend's list***searchvSearchMember**.setOnQueryTextListener(**new** SearchView.OnQueryTextListener() {  
 @Override  
 **public boolean** onQueryTextSubmit(String query) {  
 **searchvSearchMember**.clearFocus();  
 **return true**;  
 }  
  
 @Override  
 **public boolean** onQueryTextChange(String newText) {  
 **if** (!newText.isEmpty()) {  
 callSearch(newText);  
 } **else** {  
 **for** (**int** i = 0; i < **id**; i++) {  
 RelativeLayout relativeLayout = view.findViewById(R.id.***RelativeLMemberGroup*** + i);  
 relativeLayout.setVisibility(View.***VISIBLE***);  
 }  
 }  
 **return true**;  
 }  
  
 **public void** callSearch(String query) {  
 *//Do searching* **for** (**int** i = 0; i < **id**; i++) {  
 RelativeLayout relativeLayout = view.findViewById(R.id.***RelativeLMemberGroup*** + i);  
 TextView textviewUserName = view.findViewById(R.id.***TextviewMember*** + i);  
 **if** (!relativeLayout.getTag().toString().contains(query) && !textviewUserName.getText().toString().contains(query))  
 relativeLayout.setVisibility(View.***GONE***);  
 **else** relativeLayout.setVisibility(View.***VISIBLE***);  
 }  
 }  
});  
  
**btnOK**.setOnClickListener(**new** View.OnClickListener()  
  
{  
 @Override  
 **public void** onClick(View view) {  
 TextView tvMembers = *viewCreateGroup*.findViewById(R.id.***TextView\_Members***);  
 tvMembers.setText(**""**);  
 ArrayList<Integer> ListLengthWordtoSpan = **new** ArrayList<Integer>();  
 String allString = **""**;  
 **for** (**int** i = 0; i < **id**; i++) {  
 RelativeLayout relativeLayout = **linearlayoutMember**.findViewById(R.id.***RelativeLMemberGroup*** + i);  
 **final** CheckBox cbChooseMember = **linearlayoutMember**.findViewById(R.id.***addMemberButton*** + i);  
 **if** (cbChooseMember.isChecked()) {  
 ListLengthWordtoSpan.add(relativeLayout.getTag().toString().length()+ **BLANK\_SIZE**);  
 allString += relativeLayout.getTag().toString() + **" "**;  
 }  
 }  
 Spannable WordtoSpan = **new** SpannableString(allString);  
 **int** cusor = 0;  
 **for**(**int** i =0 ; i<ListLengthWordtoSpan.size();i++) {  
 cusor += ListLengthWordtoSpan.get(i);  
 WordtoSpan.setSpan(**new** BackgroundColorSpan(Color.***GRAY***),  
 cusor-ListLengthWordtoSpan.get(i), cusor-**BLANK\_SIZE**, Spanned.***SPAN\_EXCLUSIVE\_EXCLUSIVE***);  
 }  
 tvMembers.setText(WordtoSpan);  
  
 dismiss();  
 }  
});  
**btnCancel**.setOnClickListener(**new** View.OnClickListener()  
  
{  
 @Override  
 **public void** onClick(View view) {  
 dismiss();  
 }  
});  
  
LoadFriend(view);  
  
**return** builder.create();

**02. createMemberView method**

***private void*** *createMemberView(****String*** *userName,* ***String*** *ID\_User,* ***String*** *NameofUser)*

Description: Create a new view of friend in friend's list

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | userName | String | In |  | friend's username |
| 02 | ID\_User | String | In |  | friend's ID key on firebase |
| 03 | NameOfUser | String | In |  | friend's full name |

* **Implementation**

RelativeLayout relativeLayout = **new** RelativeLayout(getActivity());  
 relativeLayout.setTag(userName);  
 relativeLayout.setId(R.id.***RelativeLMemberGroup*** + **id**);  
  
 *// khởi tạo parameter cho relativeLayout (user)* RelativeLayout.LayoutParams textparams = **new** RelativeLayout.LayoutParams(  
 RelativeLayout.LayoutParams.***MATCH\_PARENT***, RelativeLayout.LayoutParams.***WRAP\_CONTENT***);  
 textparams.**topMargin** = 20;  
 textparams.**rightMargin** = 20;  
 textparams.**leftMargin** = 20;  
 *// tạo ra TextView* TextView textviewUserName = **new** TextView(getActivity());  
 textviewUserName.setId(R.id.***TextviewMember*** + **id**);textviewUserName.setBackground(getActivity().getDrawable(R.drawable.***lview\_v2***));  
 textviewUserName.setHeight(100);  
 textviewUserName.setGravity(Gravity.***CENTER***);  
 textviewUserName.setTextSize(20);  
 textviewUserName.setText(NameOfUser);  
 *// Cho TextView vào parameter vừa tạo ra* relativeLayout.addView(textviewUserName, textparams);  
  
 *// khởi tạo parameter cho relativeLayout (avatar)* RelativeLayout.LayoutParams imageparams = **new** RelativeLayout.LayoutParams(  
 RelativeLayout.LayoutParams.***WRAP\_CONTENT***, RelativeLayout.LayoutParams.***WRAP\_CONTENT***);  
 imageparams.**topMargin** = 20;  
 imageparams.**leftMargin** = 20;  
 imageparams.addRule(RelativeLayout.***ALIGN\_PARENT\_LEFT***);  
 *// tạo ra ImageView* de.hdodenhof.circleimageview.CircleImageView image = **new** de.hdodenhof.circleimageview.CircleImageView(getActivity());  
 setAvatar(ID\_User,image);  
 relativeLayout.addView(image, imageparams);  
 image.setTag(textviewUserName.getText() + **"Avatar"**);  
 image.setBorderWidth(3);  
 image.setBorderColor(Color.*parseColor*(**"#FFFFFFFF"**));  
 image.getLayoutParams().**height**=100;  
 image.getLayoutParams().**width**=100;  
 *// Cho ImageView vào parameter vừa tạo ra  
  
 // khởi tạo parameter cho relativeLayout (add)* RelativeLayout.LayoutParams addparams = **new** RelativeLayout.LayoutParams(  
 RelativeLayout.LayoutParams.***WRAP\_CONTENT***, RelativeLayout.LayoutParams.***WRAP\_CONTENT***);  
 addparams.**topMargin** = 20;  
 addparams.**rightMargin** = 30;  
 addparams.addRule(RelativeLayout.***ALIGN\_PARENT\_RIGHT***);  
 *// tạo ra TextView* CheckBox cbxAddMember = **new** CheckBox(getActivity());  
 cbxAddMember.setId(R.id.***addMemberButton*** + **id**);  
 cbxAddMember.setGravity(Gravity.***CENTER***);  
 cbxAddMember.setPaddingRelative(10, 10, 0, 0);  
 cbxAddMember.setTag(textviewUserName.getText() + **"AddMember"**); *// Cho ImageView vào parameter vừa tạo ra* relativeLayout.addView(cbxAddMember, addparams);  
  
 *// thêm thanh chat mới vào message* **linearlayoutMember**.addView(relativeLayout);  
 **id**++;

**03. mapping method**

***private void*** *mapping(****View*** *view)*

Description: Set value to all variable view by findViewById function

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | view | View | In |  | set all view to variable in group dialog |

* **Implementation**

**btnOK** = view.findViewById(R.id.***button\_OKsearchMember***);  
**btnCancel** = view.findViewById(R.id.***button\_CancelsearchMember***);  
**linearlayoutMember** = view.findViewById(R.id.***LinearlayoutsearchMember\_Message***);  
**searchvSearchMember** = view.findViewById(R.id.***searchMember\_Message***);

**04. setAvatar method**

***private void*** *setAvatar(****final******String*** *UserID,* ***final CircleImageView*** *image)*

Description: Connect to firebase and load avatar of user

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | *UserID* | *String* | In |  | ID of user |
| 02 | *image* | *CircleImageView* | In |  | avatar of user |

* **Implementation**

**if**(checkImageExist(UserID)){  
 Bitmap bitmap = loadImageFromStorage(UserID);  
 image.setImageBitmap(bitmap);  
}  
**else** {  
 **final** String AVATAR = **"avatar"**;  
 File localFile = **null**;  
 **try** {  
 localFile = File.*createTempFile*(**"images"**, **"jpg"**);  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 **final** File finalLocalFile = localFile;  
 FirebaseStorage.*getInstance*().getReference()  
 .child(AVATAR + **"/"** + UserID)  
 .getFile(localFile)  
 .addOnSuccessListener(**new** OnSuccessListener<FileDownloadTask.TaskSnapshot>()

{  
 @Override  
 **public void** onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {  
 *// Successfully downloaded data to local file* image.setImageBitmap(BitmapFactory.*decodeFile*(finalLocalFile.getAbsolutePath()));  
 saveToInternalStorage(BitmapFactory.*decodeFile*(finalLocalFile.getAbsolutePath()), UserID);  
 }  
 })  
 .addOnFailureListener(**new** OnFailureListener() {  
 @Override  
 **public void** onFailure(@NonNull Exception exception) {  
 *// Handle failed download  
 // ...* image.setImageResource(R.drawable.***ic\_user***);  
 }  
 });  
}

**05. saveToInternalStorage method**

***public void*** *saveToInternalStorage(****Bitmap*** *bitmap,* ***String*** *userId)*

Description: save Image downloaded to internal storage

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | *bitmap* | *Bitmap* | In |  | bitmap to handle image |
| 02 | *userId* | *String* | In |  | user' id on firebase |

* **Implementation**

ContextWrapper contextWrapper = **new** ContextWrapper(getApplicationContext());  
*// path to /data/data/your app/app\_data/avatar*File directory = contextWrapper.getDir(**"avatar"**, Context.***MODE\_PRIVATE***);  
  
*// Create imageDir*File mypath = **new** File(directory, userId + **".jpg"**);  
  
FileOutputStream fos = **null**;  
**try** {  
 fos = **new** FileOutputStream(mypath);  
 *// Use the compress method on the BitMap object to write image to the OutputStream* bitmap.compress(Bitmap.CompressFormat.***PNG***, 100, fos);  
} **catch** (Exception e) {  
 e.printStackTrace();  
} **finally** {  
 **try** {  
 **if** (fos != **null**) {  
 fos.close();  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
}

**06. checkImageExist method**

***private boolean*** *checkImageExist(****String*** *userId)*

Description: check image if it has been in the storage

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | userId | String | In |  | user's ID |

* **Implementation**

ContextWrapper contextWrapper = **new** ContextWrapper(getActivity().getApplicationContext());  
  
*// path to /data/data/your app/app\_data/avatar*File directory = contextWrapper.getDir(**"avatar"**, Context.***MODE\_PRIVATE***);  
  
*// Create imageDir*File mypath = **new** File(directory,userId + **".jpg"**);  
**return** mypath.exists();

**07. loadImageFromStorage method**

***private boolean*** *loadImageFromStorage(****String*** *userId)*

Description: load image from the storage

* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | userId | String | In |  | user's ID |

* **Implementation**

ContextWrapper contextWrapper = **new** ContextWrapper(getActivity().getApplicationContext());  
  
*// path to /data/data/your app/app\_data/avatar*File directory = contextWrapper.getDir(**"avatar"**, Context.***MODE\_PRIVATE***);  
  
*// Create imageDir*File mypath = **new** File(directory,userId + **".jpg"**);  
**try** {  
 **return** BitmapFactory.*decodeStream*(**new** FileInputStream(mypath));  
} **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
}  
**return null**;

**08. LoadFriend method**

***private boolean*** *LoadFriend(****final View*** *view)*

Description: connect to firebase to get list of friend

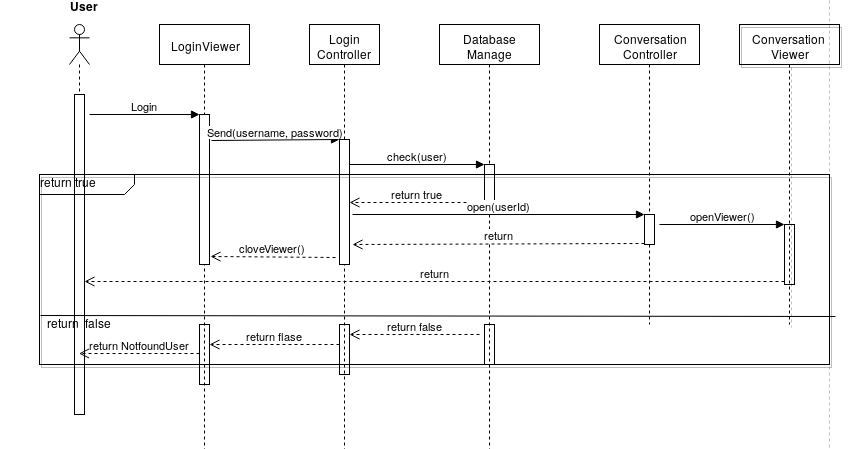
* **Parameters & return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Parameter** | **Type** | **In/out** | **Default** | **Description** |
| 01 | view | View | In |  | set all view to variable in SearchMemberDialog |

* **Implementation**

*// LoadFriend create friend chat***final** FirebaseAuth mAuth = FirebaseAuth.*getInstance*();  
**final** FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
String Username = mAuth.getCurrentUser().getDisplayName();  
String ID\_User = mAuth.getCurrentUser().getUid();  
  
TextView tvMembers = *viewCreateGroup*.findViewById(R.id.***TextView\_Members***);  
**final** String [] Member\_username = tvMembers.getText().toString().split(**" "**);  
  
**if** (Username == **null**) **return**;  
DatabaseReference ref = database.getReference(**"Chat"**).child(ID\_User);  
ref.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 **for** (DataSnapshot db : dataSnapshot.getChildren()) {  
 **final** Conversation friend = db.getValue(Conversation.**class**);  
 **final** String NameOfUser\_Friend = friend.getConversationName();  
 **if** (friend.getType().equals(**"Friend Chat"**)) {  
 DatabaseReference refUser = database.getReference(**"User"**).child(db.getKey());  
 **final** String ID\_User = db.getKey();  
 refUser.addListenerForSingleValueEvent(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot dataSnapshot) {  
 User FriendInfo = dataSnapshot.getValue(User.**class**);  
 createMemberView(NameOfUser\_Friend, ID\_User, FriendInfo.getNameOfUser());  
  
 RelativeLayout friend\_username = view.findViewById(R.id.***RelativeLMemberGroup*** + **id** - 1);  
 CheckBox cbxAddMember = view.findViewById(R.id.***addMemberButton*** + **id** - 1);  
 **for** (**int** i = 0; i < Member\_username.**length**; i++) {  
 **if** (Member\_username[i].equals(friend\_username.getTag().toString())) {  
 cbxAddMember.setChecked(**true**);  
 **break**;  
 }  
 }  
 }@Override **public void** onCancelled(DatabaseError databaseError) {}  
 });  
 }  
 }  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
});

# Diagrams

1. **Sequence Diagram**

# Packages

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Package** | **Language** | **Description** |
| 01 | activities | Java | This package contain BaseActivity, ChatActivity, MapsActivity, MessageActivity, ProfileActivity, Sign\_inActivity, Sign\_upActivity, ViewProfileActivity |
| 02 | adapters | Java | This package contain Message\_Adapter, PlaceAutocompleteAdapter |
| 03 | dialogs | Java | This package contains ChangeAvatarDialog, CreateGroupDialog, EditPasswordDialog, EditPhoneDialog, FriendRequestDialog, SearchMemberDialog |
| 04 | default | Java | This package contains Conversation, Database, DBHelper, Messages, User |

# Other considerations

*<This section provides a description of other design elements that were considered as alternatives in selection process for the above class design, i.e. a brief explanation of advantages and disadvantages of the selected package relationships and/or class implementation in comparison with others. It should be a clear answer to the question why the above class design is selected for this system, not the others.>*

# Appendix