F:\git\java\mar3\filemonitor\target\lqrwechatapp\lqrwechatapp-0.doc

0:F:\git\android\weixinlook\LQRWeChat\app\src\androidTest\java\com\lqr\wechat\ExampleInstrumentedTest.java

```
package com.lqr.wechat;

import android.content.Context;
import android.support.test.InstrumentationRegistry;
import android.support.test.runner.AndroidJUnit4;

import org.junit.Test;
import org.junit.runner.RunWith;

import static org.junit.Assert.*;

/**
 * Instrumentation test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test
    public void useAppContext() throws Exception {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getTargetContext();

        assertEquals("com.lqr.wechat", appContext.getPackageName());
    }
}
```

1:F:\git\android\weixinlook\LQRWeChat\app\src\main\AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.lqr.wechat">

    <!-- star   -->
    <!-- sd  -->
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
    <!--    -->
    <uses-feature android:name="android.hardware.camera"/>
    <uses-feature android:name="android.hardware.camera.autofocus"/>
```

```
<!-- end   -->

<!--  -->
<!-- -->
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<!--  -->
<uses-permission android:name="android.permission.FLASHLIGHT"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<!--  -->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

<!--  -->
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>

<!--  -->
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.BROADCAST_STICKY"/>

<uses-feature android:name="android.hardware.camera"/>
<uses-feature android:name="android.hardware.camera.autofocus"/>
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true"/>

<!-- SDK ,  APP  com.netease.nim.demo  -->
<!--  uses-permission  AndroidManifest  -->
<permission
    android:name="com.lqr.wechat.permission.RECEIVE_MSG"
    android:protectionLevel="signature"/>
<!--  SDK   APP  com.netease.nim.demo  -->
<uses-permission android:name="com.lqr.wechat.RECEIVE_MSG"/>

<application
    android:name=".App"
    android:allowBackup="true"
```

```
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:supportsRtl="true"
android:theme="@style/AppTheme">

<!--################################## begin##################################-->
<!-- APP key,  SDKOptions
    SDKOptions  SDKOptions  -->
<meta-data
   android:name="com.netease.nim.appKey"
   android:value="5c6b874a1803f3500e26a984f5ad33a7"/>
<!--  -->
<service
   android:name="com.netease.nimlib.service.NimService"
   android:process=":core"/>

<service
   android:name="com.netease.nimlib.service.NimService$Aux"
   android:process=":core"/>

<!-- SDK
    NimService  -->
<receiver
   android:name="com.netease.nimlib.service.NimReceiver"
   android:exported="false"
   android:process=":core">
   <intent-filter>
      <action android:name="android.intent.action.BOOT_COMPLETED"/>
      <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
   </intent-filter>
</receiver>

<!--  Receiver -->
<receiver android:name="com.netease.nimlib.service.ResponseReceiver"/>

<!--  -->
<service
   android:name="com.netease.cosine.core.CosineService"
   android:process=":cosine">
</service>

<receiver
```

```
    android:name="com.netease.cosine.target.CosineReceiver"
    android:exported="true"
    android:process=":cosine">
</receiver>

<meta-data
    android:name="com.netease.cosine.target"
    android:value=""/>
<meta-data
    android:name="com.netease.cosine.target.receiver"
    android:value="com.netease.nimlib.service.NimReceiver"/>

<!--################################### end###################################-->

<activity
    android:name=".activity.SplashActivity"
    android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>

        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
<activity
    android:name=".activity.LoginActivity"
    android:launchMode="singleTask"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.MainActivity"
    android:launchMode="singleTask"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.OtherLoginActivity"
    android:screenOrientation="portrait">
</activity>

<!---->
<activity
    android:name=".activity.SessionActivity"
    android:screenOrientation="portrait">
```

```
</activity>
<activity
    android:name=".activity.RedPacketActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.TransferActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.LocationActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.FilePreviewActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.UserInfoActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.AliasActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.FriendCirclePrivacySetActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.PostscriptActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.ImageWatchActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.FileWallActivity"
    android:screenOrientation="portrait">
</activity>
<activity
```

```
    android:name=".activity.NewFriendActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.AddFriendActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.SearchUserActivity"
    android:screenOrientation="portrait">
</activity>
<!---->
<activity
    android:name=".activity.SettingActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.NewMsgNotifySetActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.DontDistorbSetActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.CheatSetActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.PrivacySetActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.CommonSetActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.AccountAndSafeSetActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.AboutActivity"
```

```
    android:screenOrientation="portrait">
</activity>
<!---->
<activity
    android:name=".activity.CardPaketActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.MsgNotificationActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.VipCardActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.MyCouponActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.FriendsCouponActivity"
    android:screenOrientation="portrait">
</activity>

<!---->
<activity
    android:name=".activity.MyInfoActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.ShowBigImageActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.ChangeNameActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.ChangeSignatureActivity"
    android:screenOrientation="portrait">
</activity>
<activity
```

```
    android:name=".activity.QRCodeCardActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.WebViewActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.ScanActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.NearbyPerpleActivity"
    android:screenOrientation="portrait">
</activity>
<!---->
<activity
    android:name=".activity.TeamCheatCreateActvitiy"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.TeamCheatInfoActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.TeamNameSetActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.TeamCheatListActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.AllTagActvitiy"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.TeamAnnouncementEditActivity"
    android:screenOrientation="portrait">
</activity>
<activity
    android:name=".activity.TeamCheatRemoveMemberActivity"
```

```
            android:screenOrientation="portrait">
        </activity>
    </application>

</manifest>
```

2:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\AboutActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class AboutActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_about);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }
```

```
    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

3:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\AccountAndSafeSetActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class AccountAndSafeSetActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_account_and_safe_set);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
```

```
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

4:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\AddFriendActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.drawable.BitmapDrawable;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import com.lqr.wechat.AppConst;
import com.lqr.wechat.R;
import com.lqr.wechat.factory.ThreadPoolFactory;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.UserCache;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.UIUtils;
import com.lqr.wechat.view.CustomDialog;
import com.netease.nimlib.sdk.uinfo.constant.GenderEnum;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;
import cn.bingoogolapple.qrcode.zxing.QRCodeEncoder;
```

```
/**
 * @ CSDN_LQR
 * @
 */
public class AddFriendActivity extends BaseActivity {

    private Intent mIntent;

    private NimUserInfo mNimUserInfo;
    private View mQRCodeCardView;
    private CustomDialog mQRCodeCardDialog;
    private ImageView mIvHeaderQRCodeCard;
    private TextView mTvNameQRCodeCard;
    private ImageView mIvGenderQRCodeCard;
    private ImageView mIvCardQRCodeCard;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @OnClick({R.id.etContent, R.id.ivQRCordCard})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.etContent:
                mIntent = new Intent(this, SearchUserActivity.class);
                mIntent.putExtra(SearchUserActivity.SEARCH_TYPE,
SearchUserActivity.SEARCH_USER_REMOTE);
                startActivity(mIntent);
                break;
            case R.id.ivQRCordCard:
                if (mQRCodeCardView == null) {
                    mQRCodeCardView = View.inflate(AddFriendActivity.this,
R.layout.include_qrcode_card, null);
mQRCodeCardView.setBackgroundResource(R.drawable.shape_corner_rect_solid_white);
        mIvHeaderQRCodeCard = (ImageView) mQRCodeCardView.findViewById(R.id.ivHeader);
                    mTvNameQRCodeCard = (TextView)
mQRCodeCardView.findViewById(R.id.tvName);
                    mIvGenderQRCodeCard = (ImageView)
mQRCodeCardView.findViewById(R.id.ivGender);
                    mIvCardQRCodeCard = (ImageView) mQRCodeCardView.findViewById(R.id.ivCard);
                    mQRCodeCardDialog = new CustomDialog(AddFriendActivity.this, 300, 400,
mQRCodeCardView, R.style.dialog);
                }
```

```
            String avatar = mNimUserInfo.getAvatar();
            if (TextUtils.isEmpty(avatar)) {
                mIvHeaderQRCodeCard.setImageResource(R.mipmap.default_header);
            } else {
                ImageLoaderManager.LoadNetImage(avatar, mIvHeaderQRCodeCard);
            }
            mTvNameQRCodeCard.setText(mNimUserInfo.getName());
            if (mNimUserInfo.getGenderEnum() == GenderEnum.FEMALE) {
                mIvGenderQRCodeCard.setImageResource(R.mipmap.ic_gender_female);
            } else if (mNimUserInfo.getGenderEnum() == GenderEnum.MALE) {
                mIvGenderQRCodeCard.setImageResource(R.mipmap.ic_gender_male);
            } else {
                mIvGenderQRCodeCard.setVisibility(View.GONE);
            }

            Bitmap bitmap = ((BitmapDrawable)
mIvHeaderQRCodeCard.getDrawable()).getBitmap();
            showQRCordCard(bitmap);

//              ThreadPoolFactory.getNormalPool().execute(new Runnable() {
//                  @Override
//                  public void run() {
//                      OkHttpUtils.get().url(mNimUserInfo.getAvatar()).build().execute(new
BitmapCallback() {
//                          @Override
//                          public void onError(Call call, Exception e, int id) {
//                              Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
R.mipmap.default_header);
//                              showQRCordCard(bitmap);
//                          }
//
//                          @Override
//                          public void onResponse(Bitmap bitmap, int id) {
//                              showQRCordCard(bitmap);
//                          }
//                      });
//
//                  }
//              });

            mQRCodeCardDialog.show();
```

```
                break;
            }
        }

        @Override
        public void init() {
            mNimUserInfo = NimUserInfoSDK.getUser(UserCache.getAccount());
        }

        @Override
        public void initView() {
            setContentView(R.layout.activity_add_friend);
            ButterKnife.inject(this);

            initToolbar();
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
                case android.R.id.home:
                    finish();
                    break;
            }
            return super.onOptionsItemSelected(item);
        }

        private void initToolbar() {
            setSupportActionBar(mToolbar);
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setTitle("");
            mToolbar.setNavigationIcon(R.mipmap.ic_back);
        }

        private void showQRCordCard(final Bitmap bitmap) {
            ThreadPoolFactory.getNormalPool().execute(new Runnable() {
                @Override
                public void run() {
//                    final Bitmap codeWithLogo5 =
QRCodeEncoder.syncEncodeQRCode(AppConst.QRCodeCommend.ACCOUNT +
mNimUserInfo.getAccount(), UIUtils.dip2Px(200), UIUtils.getColor(R.color.transparent),
UIUtils.getColor(R.color.black0), bitmap);
```

```
            final Bitmap codeWithLogo5 =
QRCodeEncoder.syncEncodeQRCode(AppConst.QRCodeCommend.ACCOUNT +
mNimUserInfo.getAccount(), UIUtils.dip2Px(200));
            UIUtils.postTaskSafely(new Runnable() {
                @Override
                public void run() {
                    mIvCardQRCodeCard.setImageBitmap(codeWithLogo5);
                }
            });
        }
    });

  }
}
```

5:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\AliasActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;

import com.lqr.wechat.R;
import com.lqr.wechat.model.Contact;
import com.lqr.wechat.nimsdk.NimFriendSDK;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.friend.constant.FriendFieldEnum;

import java.util.HashMap;
import java.util.Map;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
```

```
 * @ CSDN_LQR
 * @
 */
public class AliasActivity extends BaseActivity {

    private String alias;
    private Contact mContact;

    public static final int REQ_CHANGE_ALIAS = 100;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.btnOk)
    Button mBtnOk;
    @InjectView(R.id.etAlias)
    EditText mEtAlias;
    @InjectView(R.id.ibClearAlias)
    ImageButton mIbClearAlias;
    @InjectView(R.id.etTag)
    EditText mEtTag;
    @InjectView(R.id.ibClearTag)
    ImageButton mIbClearTag;
    @InjectView(R.id.etPhone)
    EditText mEtPhone;
    @InjectView(R.id.ibClearPhone)
    ImageButton mIbClearPhone;
    @InjectView(R.id.etDesc)
    EditText mEtDesc;
    @InjectView(R.id.ibClearDesc)
    ImageButton mIbClearDesc;
    @InjectView(R.id.etPicture)
    EditText mEtPicture;
    @InjectView(R.id.ibClearPicture)
    ImageButton mIbClearPicture;

    @OnClick({R.id.btnOk})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.btnOk:
                saveAliasChange();
                break;
        }
```

```
}

@Override
public void init() {
    mContact = (Contact) getIntent().getSerializableExtra("contact");
    if (mContact == null) {
        interrupt();
        return;
    }
}

@Override
public void initView() {
    setContentView(R.layout.activity_alias);
    ButterKnife.inject(this);
    initToolbar();

    String alias = mContact.getFriend().getAlias();
    if (!TextUtils.isEmpty(alias)) {
        mEtAlias.setText(alias);
        mEtAlias.setSelection(alias.length());
    }
}

@Override
public void initData() {
    alias = mContact.getFriend().getAlias();
}

@Override
public void initListener() {
    mEtAlias.setOnFocusChangeListener(new View.OnFocusChangeListener() {
        @Override
        public void onFocusChange(View v, boolean hasFocus) {
            if (hasFocus) {
                mIbClearAlias.setVisibility(View.VISIBLE);
            } else {
                mIbClearAlias.setVisibility(View.GONE);
            }
        }
    });
    mIbClearAlias.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            mEtAlias.setText("");
        }
    });
}

@Override
public void onBackPressed() {
    if (!alias.equals(mEtAlias.getText().toString().trim())) {
        showMaterialDialog("", "?", "", "", new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                saveAliasChange();
                hideMaterialDialog();
            }
        }, new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                hideMaterialDialog();
            }
        });
        return;
    }
    super.onBackPressed();
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            break;
    }
    return super.onOptionsItemSelected(item);
}

private void initToolbar() {
    setSupportActionBar(mToolbar);
    getSupportActionBar().setTitle("");
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    mToolbar.setNavigationIcon(R.mipmap.ic_back);
```

```
        mBtnOk.setVisibility(View.VISIBLE);
    }

    private void saveAliasChange() {
        String alias = mEtAlias.getText().toString().trim();
        showWaitingDialog("");
        Map<FriendFieldEnum, Object> map = new HashMap<>(1);
        map.put(FriendFieldEnum.ALIAS, alias);
        NimFriendSDK.updateFriendFields(mContact.getAccount(), map, new
RequestCallback<Void>() {
            @Override
            public void onSuccess(Void param) {
                UIUtils.showToast("");
                hideWaitingDialog();
                setResult(RESULT_OK);
                finish();
            }

            @Override
            public void onFailed(int code) {
                UIUtils.showToast("" + code);
                hideWaitingDialog();
            }

            @Override
            public void onException(Throwable exception) {
                exception.printStackTrace();
                hideWaitingDialog();
            }
        });
    }
}
```

6:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\AllTagActvitiy.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
```

```
import com.lqr.wechat.R;
import com.lqr.wechat.utils.UIUtils;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class AllTagActvitiy extends BaseActivity {

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_all_tag);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        new MenuInflater(this).inflate(R.menu.menu_one_text, menu);
        menu.getItem(0).setTitle("");
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
            case R.id.itemOne:
                UIUtils.showToast("");
                break;
        }
        return super.onOptionsItemSelected(item);
    }
```

```
    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

7:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\BaseActivity.java

```
package com.lqr.wechat.activity;

import android.app.Dialog;
import android.os.Build;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.WindowManager;
import android.widget.TextView;

import com.lqr.wechat.App;
import com.lqr.wechat.R;
import com.lqr.wechat.view.CustomDialog;

import java.lang.reflect.Field;

import me.drakeet.materialdialog.MaterialDialog;


/**
 * @ CSDN_LQR
 * @ AppCompatActivity
 */
public class BaseActivity extends AppCompatActivity {

    private CustomDialog mDialogWaiting;
    private MaterialDialog mMaterialDialog;
    private boolean interrupt = false;//onCreate
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.requestWindowFeature(Window.FEATURE_NO_TITLE);

        init();
        if (interrupt) {
            finish();
            return;
        }
        initView();
        initData();
        initListener();
        App.activities.add(this);
    }

    /**
     * onCreate
     */
    public void interrupt() {
        this.interrupt = true;
    }

    /**
     *
     *
     * @param linear_bar
     */
    protected void setStatusBar(final ViewGroup linear_bar) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
            //
getWindow().addFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS);
//
getWindow().addFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_NAVIGATION);
        final int statusHeight = getStatusBarHeight();
            linear_bar.post(new Runnable() {
                @Override
                public void run() {
                    int titleHeight = linear_bar.getHeight();
                    android.widget.LinearLayout.LayoutParams params =
(android.widget.LinearLayout.LayoutParams) linear_bar.getLayoutParams();
                    params.height = statusHeight + titleHeight;
```

```
                linear_bar.setLayoutParams(params);
            }
        });
    }
}

/**
 *
 *
 * @return
 */
public int getStatusBarHeight() {
    try {
        Class<?> c = Class.forName("com.android.internal.R$dimen");
        Object obj = c.newInstance();
        Field field = c.getField("status_bar_height");
        int x = Integer.parseInt(field.get(obj).toString());
        return getResources().getDimensionPixelSize(x);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return 0;
}

public void init() {
}

public void initView() {
}

public void initData() {
}

public void initListener() {
}

@Override
protected void onDestroy() {
    super.onDestroy();
    App.activities.remove(this);
}
```

```
    /**
     *
     */
    public Dialog showWaitingDialog(String tip) {
        hideWaitingDialog();
        View view = View.inflate(this, R.layout.dialog_waiting, null);
        if (!TextUtils.isEmpty(tip))
            ((TextView) view.findViewById(R.id.tvTip)).setText(tip);
        mDialogWaiting = new CustomDialog(this, view, R.style.dialog);
        mDialogWaiting.show();
        mDialogWaiting.setCancelable(false);
        return mDialogWaiting;
    }

    /**
     *
     */
    public void hideWaitingDialog() {
        if (mDialogWaiting != null) {
            mDialogWaiting.dismiss();
            mDialogWaiting = null;
        }
    }

    /**
     * MaterialDialog
     */
    public MaterialDialog showMaterialDialog(String tip, String message, String positiveText, String
negativeText, View.OnClickListener positiveButtonClickListener, View.OnClickListener
negativeButtonClickListener) {
        hideMaterialDialog();
        mMaterialDialog = new MaterialDialog(this);
        if (!TextUtils.isEmpty(tip)) {
            mMaterialDialog.setTitle(tip);
        }
        if (!TextUtils.isEmpty(message)) {
            mMaterialDialog.setMessage(message);
        }
        if (!TextUtils.isEmpty(positiveText)) {
            mMaterialDialog.setPositiveButton(positiveText, positiveButtonClickListener);
        }
        if (!TextUtils.isEmpty(negativeText)) {
```

```
            mMaterialDialog.setNegativeButton(negativeText, negativeButtonClickListener);
        }
        mMaterialDialog.show();
        return mMaterialDialog;
    }

    /**
     * MaterialDialog
     */
    public void hideMaterialDialog() {
        if (mMaterialDialog != null) {
            mMaterialDialog.dismiss();
            mMaterialDialog = null;
        }
    }

}
```

8:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\CardPaketActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class CardPaketActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
```

```
@OnClick({R.id.cvVipCard, R.id.cvFriendsCoupon, R.id.cvMyCoupon})
public void click(View view) {
    switch (view.getId()) {
        case R.id.cvVipCard:
            startActivity(new Intent(this, VipCardActivity.class));
            break;
        case R.id.cvFriendsCoupon:
            startActivity(new Intent(this, FriendsCouponActivity.class));
            break;
        case R.id.cvMyCoupon:
            startActivity(new Intent(this, MyCouponActivity.class));
            break;
    }
}

@Override
public void initView() {
    setContentView(R.layout.activity_card_packet);
    ButterKnife.inject(this);
    initToolbar();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    new MenuInflater(this).inflate(R.menu.menu_one_text, menu);
    menu.getItem(0).setTitle("");
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            break;
        case R.id.itemOne:
            startActivity(new Intent(this, MsgNotificationActivity.class));
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

```
    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

9:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\ChangeNameActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.uinfo.constant.UserInfoFieldEnum;

import java.util.HashMap;
import java.util.Map;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class ChangeNameActivity extends BaseActivity {

    private String mName;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
```

```
@InjectView(R.id.btnOk)
Button mBtnOk;
@InjectView(R.id.etName)
EditText mEtName;

@OnClick({R.id.btnOk})
public void click(View view) {
    switch (view.getId()) {
        case R.id.btnOk:
            String name = mEtName.getText().toString();
            if (TextUtils.isEmpty(name.trim())) {
                showMaterialDialog("", "", "", "", new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        hideMaterialDialog();
                    }
                }, null);
            } else {
                showWaitingDialog("");
                Map<UserInfoFieldEnum, Object> fields = new HashMap<>(1);
                fields.put(UserInfoFieldEnum.Name, name);
                NimUserInfoSDK.updateUserInfo(fields, new RequestCallbackWrapper<Void>() {
                    @Override
                    public void onResult(int code, Void result, Throwable exception) {
                        hideWaitingDialog();
                        finish();
                    }
                });
            }
            break;
    }
}

@Override
public void init() {
    mName = getIntent().getStringExtra("name");
}

@Override
public void initView() {
    setContentView(R.layout.activity_change_name);
    ButterKnife.inject(this);
```

```
        initToolbar();
        mEtName.setText(mName);
        mEtName.setSelection(mName.length());
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("");
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
        mBtnOk.setVisibility(View.VISIBLE);
    }

}
```

10:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\ChangeSignatureActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.uinfo.constant.UserInfoFieldEnum;
```

```
import java.util.HashMap;
import java.util.Map;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class ChangeSignatureActivity extends BaseActivity {

    private String mSignature;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.btnOk)
    Button mBtnOk;
    @InjectView(R.id.etName)
    EditText mEtName;
    @InjectView(R.id.tvCount)
    TextView mTvCount;

    @OnClick({R.id.btnOk})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.btnOk:
                String name = mEtName.getText().toString();
                showWaitingDialog("");
                Map<UserInfoFieldEnum, Object> fields = new HashMap<>(1);
                fields.put(UserInfoFieldEnum.SIGNATURE, name);
                NimUserInfoSDK.updateUserInfo(fields, new RequestCallbackWrapper<Void>() {
                    @Override
                    public void onResult(int code, Void result, Throwable exception) {
                        hideWaitingDialog();
                        finish();
                    }
                });
                break;
        }
```

```
}

@Override
public void init() {
    mSignature = getIntent().getStringExtra("signature");
}

@Override
public void initView() {
    setContentView(R.layout.activity_change_signature);
    ButterKnife.inject(this);
    initToolbar();
    mEtName.setText(mSignature);
    mEtName.setSelection(mSignature.length());
    mTvCount.setText(String.valueOf(30 - mEtName.getText().toString().length()));
}

@Override
public void initListener() {
    mEtName.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            mTvCount.setText(String.valueOf(30 - mEtName.getText().toString().length()));
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    });
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
```

```
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("");
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
        mBtnOk.setVisibility(View.VISIBLE);
    }

}
```

11:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\CheatSetActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class CheatSetActivity extends BaseActivity {

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_cheat_set);
        ButterKnife.inject(this);
        ButterKnife.inject(this);
        initToolbar();
```

```
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

12:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\CommonSetActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */

public class CommonSetActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
```

```
    public void initView() {
        setContentView(R.layout.activity_common_set);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

13:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\DontDistorbSetActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class DontDistorbSetActivity extends BaseActivity {
```

```
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_dont_distorb_set);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

14:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\FilePreviewActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
```

```
import com.lqr.wechat.R;
import com.lqr.wechat.utils.FileIconUtils;
import com.lqr.wechat.utils.FileOpenUtils;
import com.lqr.wechat.utils.FileUtils;
import com.lqr.wechat.utils.MimeTypeUtils;
import com.netease.nimlib.sdk.msg.attachment.FileAttachment;
import com.netease.nimlib.sdk.msg.model.IMMessage;
import com.zhy.http.okhttp.OkHttpUtils;
import com.zhy.http.okhttp.callback.FileCallBack;

import java.io.File;

import butterknife.ButterKnife;
import butterknife.InjectView;
import okhttp3.Call;
import okhttp3.Request;

/**
 * @ CSDN_LQR
 * @
 */
public class FilePreviewActivity extends BaseActivity {

    private Intent mIntent;
    private IMMessage mMessage;
    private FileAttachment mFa;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.ivPic)
    ImageView mIvPic;
    @InjectView(R.id.tvName)
    TextView mTvName;
    @InjectView(R.id.pbFile)
    ProgressBar mPbFile;
    @InjectView(R.id.btnOpen)
    Button mBtnOpen;//

    @Override
    public void init() {
        mIntent = getIntent();
        mMessage = (IMMessage) mIntent.getSerializableExtra("message");
```

```
        if (mMessage == null) {
            interrupt();
            return;
        }
        mFa = (FileAttachment) mMessage.getAttachment();
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_file_preview);
        ButterKnife.inject(this);
        initToolbar();

        setFileInfo();
    }

    @Override
    public void initData() {
        //
//        if (TextUtils.isEmpty(mFa.getPath())) {
//            downloadFile();
//        }
    }

    @Override
    public void initListener() {
        mBtnOpen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mBtnOpen.getText().equals("")) {
                    downloadFile();
                } else {
                    //
                    FileOpenUtils.openFile(FilePreviewActivity.this, mFa.getPath(),
MimeTypeUtils.getMimeType(mFa.getDisplayName()));
                }
            }
        });
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
```

```
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }

    private void setFileInfo() {
        mIvPic.setImageResource(FileIconUtils.getFileIconResId(mFa.getExtension()));
        mTvName.setText(mFa.getDisplayName());
        mPbFile.setVisibility(View.GONE);
        mBtnOpen.setVisibility(View.VISIBLE);
        if (TextUtils.isEmpty(mFa.getPath())) {
            mBtnOpen.setText("");
        } else {
            mBtnOpen.setText("");
        }
    }

    //
    private void downloadFile() {
        OkHttpUtils.get().url(mFa.getUrl()).build().execute(new
FileCallBack(FileUtils.getDirFromPath(mFa.getPathForSave()),
FileUtils.getFileNameFromPath(mFa.getPathForSave())) {


            @Override
            public void onError(Call call, Exception e, int id) {
                mIvPic.setImageResource(R.mipmap.default_img_failed);
                mTvName.setText("");
                mPbFile.setVisibility(View.GONE);
                mBtnOpen.setVisibility(View.GONE);
            }
```

```
            @Override
            public void onResponse(File response, int id) {
                setFileInfo();
            }

            @Override
            public void inProgress(float progress, long total, int id) {
                super.inProgress(progress, total, id);
                mPbFile.setMax((int) total);
                mPbFile.setProgress((int) (progress * 100));
            }

            @Override
            public void onBefore(Request request, int id) {
                super.onBefore(request, id);
                mPbFile.setVisibility(View.VISIBLE);
                mBtnOpen.setVisibility(View.GONE);
            }
        });
    }
}
```

15:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\FileWallActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.graphics.Bitmap;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;

import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
```

```
import com.lqr.recyclerview.LQRRecyclerView;
import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimHistorySDK;
import com.lqr.wechat.nimsdk.NimMessageSDK;
import com.lqr.wechat.utils.Bimp;
import com.lqr.wechat.utils.LogUtils;
import com.netease.nimlib.sdk.AbortableFuture;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.msg.MessageBuilder;
import com.netease.nimlib.sdk.msg.attachment.FileAttachment;
import com.netease.nimlib.sdk.msg.constant.MsgTypeEnum;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.msg.model.IMMessage;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @ ()
 */
public class FileWallActivity extends BaseActivity {

    public static final int CHECK_RESULT_CODE = 100;

    private boolean mIsEditMode = false;//
    private IMMessage mCurrentMsg;
    private String mAccount;
    private SessionTypeEnum mSessionType;

    private List<IMMessage> mData = new ArrayList<>();
    private LQRAdapterForRecyclerView<IMMessage> mAdapter;

    private List<IMMessage> mCheckedData = new ArrayList<>();

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.cvFile)
```

```
LQRRecyclerView mCvFile;

@InjectView(R.id.llBottom)
LinearLayout mLlBottom;
@InjectView(R.id.rlShare)
RelativeLayout mRlShare;
@InjectView(R.id.rlCollect)
RelativeLayout mRlCollect;
@InjectView(R.id.rlDel)
RelativeLayout mRlDel;

@InjectView(R.id.btnShare)
Button mBtnShare;
@InjectView(R.id.btnCollect)
Button mBtnCollect;
@InjectView(R.id.btnDel)
Button mBtnDel;


@Override
public void init() {
    Intent intent = getIntent();
    mAccount = intent.getStringExtra("account");
    mCurrentMsg = (IMMessage) intent.getSerializableExtra("currentMsg");
    mSessionType = (SessionTypeEnum) intent.getSerializableExtra("sessionType");
}

@Override
public void initView() {
    setContentView(R.layout.activity_file_wall);
    ButterKnife.inject(this);
    initToolbar();
}

@Override
public void initData() {
    setAdapter();
    loadLocalImageMessage();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
```

```
    new MenuInflater(this).inflate(R.menu.menu_one_text, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            onBackPressed();
            break;
        case R.id.itemOne:
            if (mIsEditMode) {//
                //
                quitEditMode();
                item.setTitle("");
            } else {//
                //
                enterEditMode();
                item.setTitle("");
            }
            updateToolbarTitleAndBottom();
            break;
    }
    return super.onOptionsItemSelected(item);
}

private void enterEditMode() {
    mIsEditMode = true;
    mCheckedData.clear();
    setAdapter();
    mLlBottom.setVisibility(View.VISIBLE);
}

private void quitEditMode() {
    mIsEditMode = false;
    setAdapter();
    mLlBottom.setVisibility(View.GONE);
    mCvFile.setAdapter(mAdapter);
}

private void initToolbar() {
    setSupportActionBar(mToolbar);
```

```
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
        getSupportActionBar().setTitle("");
    }

    private void loadLocalImageMessage() {
        IMMessage anchor = MessageBuilder.createEmptyMessage(mAccount, mSessionType, 0);
        //100
        NimHistorySDK.queryMessageListByType(MsgTypeEnum.image, anchor,
Integer.MAX_VALUE).setCallback(new RequestCallback<List<IMMessage>>() {
            @Override
            public void onSuccess(List<IMMessage> result) {
                Collections.reverse(result);
                mAdapter.addMoreData(result);

                //
                if (mCurrentMsg != null)
                    for (int i = 0; i < result.size(); i++) {
                        if (result.get(i).getUuid().equals(mCurrentMsg.getUuid())) {
                            mCvFile.moveToPosition(i);
                            break;
                        }
                    }
            }

            @Override
            public void onFailed(int code) {
                LogUtils.e("code = " + code);
            }

            @Override
            public void onException(Throwable exception) {
                exception.printStackTrace();
            }
        });
    }

    private void setAdapter() {
        if (mAdapter == null) {
            mAdapter = new LQRAdapterForRecyclerView<IMMessage>(this, R.layout.item_file_wall,
mData) {
                @Override
```

```
            public void convert(LQRViewHolderForRecyclerView helper, final IMMessage item, int
position) {

                setImage(helper, item, position);

                helper.setViewVisibility(R.id.cb, mIsEditMode ? View.VISIBLE : View.GONE)
                    .setViewVisibility(R.id.vMask, mIsEditMode ? View.VISIBLE : View.GONE)
                    .getView(R.id.root).setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View v) {
                    Intent intent = new Intent(FileWallActivity.this, ImageWatchActivity.class);
                    intent.putExtra("account", mAccount);
                    intent.putExtra("sessionType", mSessionType);
                    intent.putExtra("message", item);
                    intent.putExtra("isEditMode", mIsEditMode);
                    startActivityForResult(intent, CHECK_RESULT_CODE);
                  }
                });

                ((CheckBox) helper.getView(R.id.cb)).setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
                  @Override
                  public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
                    if (isChecked) {
                      mCheckedData.add(item);
                    } else {
                      mCheckedData.remove(item);
                    }
                    updateToolbarTitleAndBottom();
                  }
                });
              }
            };
            mCvFile.setAdapter(mAdapter);
        } else {
            mAdapter.notifyDataSetChanged();
        }
    }

    private void updateToolbarTitleAndBottom() {
        if (mIsEditMode) {
```

```
        mToolbar.setTitle("" + mCheckedData.size() + "");
        mBtnShare.setEnabled(mCheckedData.size() > 0 ? true : false);
        mBtnCollect.setEnabled(mCheckedData.size() > 0 ? true : false);
        mBtnDel.setEnabled(mCheckedData.size() > 0 ? true : false);
    } else
        mToolbar.setTitle("");

}

private void setImage(LQRViewHolderForRecyclerView helper, IMMessage item, int position) {
    final ImageView iv = helper.getView(R.id.ivShowPic);
    final FileAttachment fa = (FileAttachment) mAdapter.getItem(position).getAttachment();

    //
    if (fa.getThumbPath() == null) {
        AbortableFuture abortableFuture = NimMessageSDK.downloadAttachment(item, true);
        abortableFuture.setCallback(new RequestCallback() {
            @Override
            public void onSuccess(Object param) {
                Bitmap bitmap = Bimp.getLoacalBitmap(fa.getThumbPath());
                if (bitmap != null) {
                    iv.setImageBitmap(bitmap);
                }
            }

            @Override
            public void onFailed(int code) {

            }

            @Override
            public void onException(Throwable exception) {

            }
        });
    } else {
        Bitmap bitmap = Bimp.getLoacalBitmap(fa.getThumbPath());
        if (bitmap != null) {
            iv.setImageBitmap(bitmap);
        }
    }
}
```

```
}
```

16:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\FriendCirclePrivacySetActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class FriendCirclePrivacySetActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_friends_circle_privacy_set);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
```

```
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

17:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\FriendsCouponActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;

import com.lqr.wechat.R;
import com.lqr.wechat.view.CustomDialog;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @ -
 */
public class FriendsCouponActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    private CustomDialog mDialog;

    @Override
    public void initView() {
        setContentView(R.layout.activity_friends_coupon);
        ButterKnife.inject(this);
        initToolbar();
        showTipDialog();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
```

```
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }

    private void showTipDialog() {
        View view = View.inflate(this, R.layout.dialog_tip_friends_coupon, null);
        mDialog = new CustomDialog(this, view, R.style.dialog);
        mDialog.setCancelable(false);
        mDialog.show();
        view.findViewById(R.id.tvOk).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mDialog.dismiss();
                mDialog = null;
            }
        });
    }
}
```

18:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\ImageWatchActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.support.v4.view.PagerAdapter;
import android.support.v4.view.ViewPager;
import android.text.TextUtils;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.RelativeLayout;

import com.bm.library.PhotoView;
```

```
import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimHistorySDK;
import com.lqr.wechat.utils.Bimp;
import com.lqr.wechat.utils.FileUtils;
import com.lqr.wechat.utils.LogUtils;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.ResponseCode;
import com.netease.nimlib.sdk.msg.MessageBuilder;
import com.netease.nimlib.sdk.msg.attachment.ImageAttachment;
import com.netease.nimlib.sdk.msg.constant.MsgTypeEnum;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.msg.model.IMMessage;
import com.zhy.http.okhttp.OkHttpUtils;
import com.zhy.http.okhttp.callback.FileCallBack;

import java.io.File;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;
import okhttp3.Call;

/**
 * @ CSDN_LQR
 * @ ()
 */
public class ImageWatchActivity extends BaseActivity {

    private String mAccount;
    private SessionTypeEnum mSessionType;
    private IMMessage mOriMessage;
    private IMMessage mAnchor;
    private List<IMMessage> mData = new ArrayList<>();
    private PhotoViewPagerAdapter mAdapter;

    private boolean isFirstLoad = true;
    private int mCurrentItem;
    private boolean mIsEditMode;//
```

```
@InjectView(R.id.root)
RelativeLayout mRlRoot;
@InjectView(R.id.btnWatchOrigImage)
Button mBtnWatchOrigImage;
@InjectView(R.id.vpImage)
ViewPager mVpImage;
@InjectView(R.id.pbLoading)
ProgressBar mPbLoading;
@InjectView(R.id.ivShowPic)
ImageView mIvShowPic;


@OnClick({R.id.ibWall, R.id.btnWatchOrigImage})
public void click(View view) {
    switch (view.getId()) {
        case R.id.btnWatchOrigImage:
            loadOrignImage();
            break;
        case R.id.ibWall:
            //
            Intent intent = new Intent(this, FileWallActivity.class);
            intent.putExtra("account", mAccount);
            intent.putExtra("sessionType", mSessionType);
            intent.putExtra("currentMsg", mData.get(mVpImage.getCurrentItem()));
            startActivity(intent);
            break;
    }
}

@Override
public void init() {
    Intent intent = getIntent();
    mAccount = intent.getStringExtra("account");
    mIsEditMode = intent.getBooleanExtra("isEditMode", false);
    mSessionType = (SessionTypeEnum) intent.getSerializableExtra("sessionType");
    mOriMessage = (IMMessage) intent.getSerializableExtra("message");
    mAnchor = MessageBuilder.createEmptyMessage(mAccount, mSessionType, 0);
    loadPreImage();
}

@Override
```

```
public void initView() {
    setContentView(R.layout.activity_image_watch);
    ButterKnife.inject(this);

    mAdapter = new PhotoViewPagerAdapter();
    mVpImage.setAdapter(mAdapter);
}

@Override
public void initListener() {
    mVpImage.setOnPageChangeListener(new ViewPager.OnPageChangeListener() {
        @Override
        public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {

        }

        @Override
        public void onPageSelected(int position) {
            if (position == 0) {
                loadPreImage();
            }
            showBtnWatchOrignImage();
        }

        @Override
        public void onPageScrollStateChanged(int state) {

        }
    });
}

/**
 * “”
 */
private void showBtnWatchOrignImage() {
    //
    int currentItem = mVpImage.getCurrentItem();
    ImageAttachment ia = (ImageAttachment) mData.get(currentItem).getAttachment();
    if (!TextUtils.isEmpty(ia.getPath())) {
        showWatchOrignBtn(false);
    } else {
        showWatchOrignBtn(true);
```

```
        }
    }

    /**
     * 10
     */
    private void loadPreImage() {

        LogUtils.sf("loadPreImage");

        NimHistorySDK.queryMessageListByType(MsgTypeEnum.image, mAnchor,
10).setCallback(new RequestCallbackWrapper<List<IMMessage>>() {
            @Override
            public void onResult(int code, List<IMMessage> result, Throwable exception) {
                if (code != ResponseCode.RES_SUCCESS || exception != null || result == null ||
result.size() == 0) {
                    return;
                }

                Collections.reverse(result);

                //0
                mAnchor = result.get(0);

                //1
                List<IMMessage> tmpList = new ArrayList<>();
                for (int i = 0; i < result.size(); i++) {
                    IMMessage message = result.get(i);
                    if (message.getMsgType() == MsgTypeEnum.image) {
                        tmpList.add(message);
                    }
                }

                //2
                if (tmpList.isEmpty()) {
                    loadPreImage();
                } else {
                    mData.addAll(0, tmpList);

                    //3
                    if (isFirstLoad)
                        for (int i = 0; i < result.size(); i++) {
```

```
                    IMMessage message = result.get(i);
                    if (message.isTheSame(mOriMessage)) {
                        mCurrentItem = i;
                        LogUtils.sf(":" + mCurrentItem);
                        break;
                    }
                }
                mAdapter.notifyDataSetChanged();
                UIUtils.postTaskSafely(new Runnable() {
                    @Override
                    public void run() {
                        if (isFirstLoad) {
                            mVpImage.setCurrentItem(mCurrentItem, false);
                        } else {
                            mVpImage.setCurrentItem(mCurrentItem + mData.size(), false);
                        }
                        showBtnWatchOrignImage();
                    }
                });
                isFirstLoad = false;

                //
                if (mData.size() == 1) {
                    loadPreImage();
                }
            }
        }
    });
}

class PhotoViewPagerAdapter extends PagerAdapter {

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

    @Override
    public boolean isViewFromObject(View view, Object object) {
        return view == object;
    }
```

```
        @Override
        public Object instantiateItem(ViewGroup container, int position) {
            PhotoView pv = new PhotoView(ImageWatchActivity.this);
            pv.enable();//
            pv.setScaleType(ImageView.ScaleType.CENTER_INSIDE);

            ImageAttachment ia = (ImageAttachment) mData.get(position).getAttachment();
            //
            if (TextUtils.isEmpty(ia.getPath())) {
                //
                if (!TextUtils.isEmpty(ia.getThumbPath())) {
                    pv.setImageBitmap(Bimp.getLoacalBitmap(ia.getThumbPath()));
                }
                //
//                ImageLoaderManager.LoadNetImage(ia.getUrl(), pv);
            } else {
                pv.setImageBitmap(Bimp.getLoacalBitmap(ia.getPath()));
            }

            container.addView(pv);
            return pv;
        }

        @Override
        public void destroyItem(ViewGroup container, int position, Object object) {
            container.removeView((View) object);
        }

        private int mChildCount = 0;

        @Override
        public void notifyDataSetChanged() {
            mChildCount = getCount();
            super.notifyDataSetChanged();
        }

        @Override
        public int getItemPosition(Object object) {
            if (mChildCount > 0) {
                mChildCount--;
                return POSITION_NONE;
            }
```

```
            return super.getItemPosition(object);
        }
    }

    /**
     * “”
     */
    public void showWatchOrignBtn(boolean show) {
        mBtnWatchOrigImage.setVisibility(show ? View.VISIBLE : View.GONE);
    }

    /**
     *
     */
    public void loadOrignImage() {
        mPbLoading.setVisibility(View.VISIBLE);
        mBtnWatchOrigImage.setEnabled(true);

        int currentItem = mVpImage.getCurrentItem();
        ImageAttachment ia = (ImageAttachment) mData.get(currentItem).getAttachment();

        //
        OkHttpUtils.get().url(ia.getUrl()).build().execute(new
FileCallBack(FileUtils.getDirFromPath(ia.getPathForSave()),
FileUtils.getFileNameFromPath(ia.getPathForSave())) {
            @Override
            public void onError(Call call, Exception e, int i) {
                mBtnWatchOrigImage.setEnabled(true);
                showWatchOrignBtn(true);
//                  mBtnWatchOrigImage.setVisibility(View.GONE);
                UIUtils.showToast("");
            }

            @Override
            public void onResponse(File file, int i) {
                mBtnWatchOrigImage.setEnabled(true);
                showWatchOrignBtn(false);
                mPbLoading.setVisibility(View.GONE);
                UIUtils.postTaskSafely(new Runnable() {
                    @Override
                    public void run() {
                        mAdapter.notifyDataSetChanged();
```

```
                }
            });
        }
    });
}

}
```

19:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\LocationActivity.java

```
package com.lqr.wechat.activity;

import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
import com.lqr.recyclerview.LQRRecyclerView;
import com.lqr.wechat.R;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class LocationActivity extends BaseActivity {

    private List<String> mData = new ArrayList<>();

    @InjectView(R.id.cvLocation)
    LQRRecyclerView mCvLocation;
    private LQRAdapterForRecyclerView<String> mAdapter;

    @Override
    public void initView() {
        setContentView(R.layout.activity_location);
        ButterKnife.inject(this);
    }

    @Override
```

```
    public void initData() {
        for (int i = 0; i < 100; i++) {
            mData.add("item " + i);
        }
        setAdapter();
    }

    private void setAdapter() {
        mAdapter = new LQRAdapterForRecyclerView<String>(this, R.layout.item_contact_cv,
mData) {
            @Override
            public void convert(LQRViewHolderForRecyclerView helper, String item, int position) {
                helper.setText(R.id.tvName, item);
            }
        };
        mCvLocation.setAdapter(mAdapter);
    }
}
```

20:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\LoginActivity.java

```
package com.lqr.wechat.activity;


import android.content.Intent;
import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.lqr.wechat.R;
import com.lqr.wechat.model.UserCache;
import com.lqr.wechat.nimsdk.NimAccountSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.AbortableFuture;
import com.netease.nimlib.sdk.RequestCallback;
```

```
import com.netease.nimlib.sdk.auth.LoginInfo;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;
import me.drakeet.materialdialog.MaterialDialog;

/**
 * @ CSDN_LQR
 * @
 */
public class LoginActivity extends BaseActivity {

    private String mUsername;
    private String mPassword;
    private String mToken;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @InjectView(R.id.etPhone)
    EditText mEtPhone;
    @InjectView(R.id.etPwd)
    EditText mEtPwd;
    @InjectView(R.id.vLinePhone)
    View mVLinePhone;
    @InjectView(R.id.vLinePwd)
    View mVLinePwd;

    @InjectView(R.id.btnLogin)
    Button mBtnLogin;
    private AbortableFuture<LoginInfo> mLoginRequest;

    @OnClick(R.id.tvOtherLogin)
    public void otherLogin() {
        startActivity(new Intent(this, OtherLoginActivity.class));
    }

    @Override
```

```
    public void initView() {
        setContentView(R.layout.activity_login);
        ButterKnife.inject(this);
        initToolbar();

        if (!TextUtils.isEmpty(mEtPhone.getText().toString()) &&
!TextUtils.isEmpty(mEtPwd.getText().toString())) {
            mBtnLogin.setEnabled(true);
        }
    }

    @Override
    public void initListener() {
        /*------------------  begin ------------------*/
        mEtPhone.setOnFocusChangeListener(new View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                if (hasFocus) {
                    mVLinePhone.setBackgroundColor(UIUtils.getColor(R.color.green0));
                } else {
                    mVLinePhone.setBackgroundColor(UIUtils.getColor(R.color.line));
                }
            }
        });
        mEtPwd.setOnFocusChangeListener(new View.OnFocusChangeListener() {
            @Override
            public void onFocusChange(View v, boolean hasFocus) {
                if (hasFocus) {
                    mVLinePwd.setBackgroundColor(UIUtils.getColor(R.color.green0));
                } else {
                    mVLinePwd.setBackgroundColor(UIUtils.getColor(R.color.line));
                }
            }
        });
        mEtPhone.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
```

```
            if (!TextUtils.isEmpty(mEtPhone.getText().toString()) &&
!TextUtils.isEmpty(mEtPwd.getText().toString())) {
                mBtnLogin.setEnabled(true);
            } else {
                mBtnLogin.setEnabled(false);
            }
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    });
    mEtPwd.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            if (!TextUtils.isEmpty(mEtPhone.getText().toString()) &&
!TextUtils.isEmpty(mEtPwd.getText().toString())) {
                mBtnLogin.setEnabled(true);
            } else {
                mBtnLogin.setEnabled(false);
            }
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    });
    /*------------------  end ------------------*/

    mBtnLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            doLogin();
            mBtnLogin.setEnabled(false);
        }
```

```
        });
        super.initListener();
    }

    /**
     * ToolBar
     */
    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("");
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }

    /**
     *
     */
    public void doLogin() {
        showWaitingDialog("...");
        mUsername = mEtPhone.getText().toString().trim();
        mPassword = mEtPwd.getText().toString().trim();
        //
        if (TextUtils.isEmpty(mUsername) || TextUtils.isEmpty(mPassword)) {
            UIUtils.showToast("");
            return;
        }
        //token(MD5md5token)
//        mToken = MD5Utils.decode16(mPassword);
        mToken = mPassword;

        //
        mLoginRequest = NimAccountSDK.login(mUsername, mToken, new
RequestCallback<LoginInfo>() {
            @Override
            public void onSuccess(LoginInfo param) {
                onLoginDone();

                //
                UserCache.setAccount(mUsername);
                //APP
                NimAccountSDK.saveUserAccount(mUsername);
                NimAccountSDK.saveUserToken(mToken);
```

```
            //
            List<String> list = new ArrayList<String>();
            list.add(UserCache.getAccount());
            NimUserInfoSDK.getUserInfosFormServer(list, null);

            //
            Intent intent = new Intent(LoginActivity.this, MainActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
            finish();
          }

          @Override
          public void onFailed(int code) {
            onLoginDone();
            if (code == 302 || code == 404) {
              MaterialDialog materialDialog = showMaterialDialog("", "", "", "", new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                  hideMaterialDialog();
                }
              }, null);
              TextView tv = new TextView(LoginActivity.this);
              tv.setText("");
              tv.setTextColor(UIUtils.getColor(R.color.black0));
              tv.setPadding(0, UIUtils.dip2Px(15), 0, UIUtils.dip2Px(18));
              materialDialog.setContentView(tv);
//              UIUtils.showToast("");
            } else {
              UIUtils.showToast(": " + code);
            }
          }

          @Override
          public void onException(Throwable exception) {
            onLoginDone();
            UIUtils.showToast("");
          }
        });
    }
```

```
    private void onLoginDone() {
        hideWaitingDialog();
        mLoginRequest = null;
        mBtnLogin.setEnabled(true);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            //ToolBar
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }


}
```

21:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\MainActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Color;
import android.support.v4.view.ViewPager;
import android.support.v7.widget.Toolbar;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.TextView;

import com.lqr.wechat.AppConst;
import com.lqr.wechat.R;
import com.lqr.wechat.adapter.MainPagerAdapter;
import com.lqr.wechat.broadcast.AuthBroadcastReceiver;
```

```
import com.lqr.wechat.factory.PopupWindowFactory;
import com.lqr.wechat.fragment.BaseFragment;
import com.lqr.wechat.fragment.ContactsFragment;
import com.lqr.wechat.fragment.DiscoveryFragment;
import com.lqr.wechat.fragment.MeFragment;
import com.lqr.wechat.fragment.MessageFragment;
import com.lqr.wechat.nimsdk.NimAccountSDK;
import com.lqr.wechat.nimsdk.NimFriendSDK;
import com.lqr.wechat.nimsdk.NimSystemSDK;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.nimsdk.custom.CustomAttachParser;
import com.lqr.wechat.utils.LogUtils;
import com.lqr.wechat.utils.StringUtils;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.InvocationFuture;
import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.StatusCode;
import com.netease.nimlib.sdk.friend.model.FriendChangedNotify;
import com.netease.nimlib.sdk.msg.MsgService;
import com.netease.nimlib.sdk.msg.constant.SystemMessageType;
import com.netease.nimlib.sdk.msg.model.SystemMessage;
import com.netease.nimlib.sdk.team.model.Team;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class MainActivity extends BaseActivity {
```

```
public static final int REQ_CLEAR_UNREAD = 100;

private int exit = 0;
private MessageFragment mMessageFragment;
private ContactsFragment mContactsFragment;
private DiscoveryFragment mDiscoveryFragment;
private MeFragment mMeFragment;
private List<BaseFragment> mFragments;

private PopupWindow mPopupWindow;

private List<SystemMessage> items = new ArrayList<>();//
private static final boolean MERGE_ADD_FRIEND_VERIFY = true; //
private Set<String> addFriendVerifyRequestAccounts = new HashSet<>(); //

private AuthBroadcastReceiver mAuthBroadcastReceiver;
private Observer<StatusCode> mOnlineStatusObserver;

@InjectView(R.id.toolbar)
Toolbar mToolbar;

@InjectView(R.id.vpContent)
ViewPager mVpContent;

//
@InjectView(R.id.llButtom)
LinearLayout mLlBottom;

@InjectView(R.id.tvMessageNormal)
TextView mTvMessageNormal;
@InjectView(R.id.tvMessagePress)
TextView mTvMessagePress;
@InjectView(R.id.tvMessageTextNormal)
TextView mTvMessageTextNormal;
@InjectView(R.id.tvMessageTextPress)
TextView mTvMessageTextPress;
@InjectView(R.id.tvMessageCount)
public TextView mTvMessageCount;

@InjectView(R.id.tvContactsNormal)
TextView mTvContactsNormal;
@InjectView(R.id.tvContactsPress)
```

```
TextView mTvContactsPress;
@InjectView(R.id.tvContactsTextNormal)
TextView mTvContactsTextNormal;
@InjectView(R.id.tvContactsTextPress)
TextView mTvContactsTextPress;
@InjectView(R.id.tvContactCount)
public TextView mTvContactCount;

@InjectView(R.id.tvDiscoveryNormal)
TextView mTvDiscoveryNormal;
@InjectView(R.id.tvDiscoveryPress)
TextView mTvDiscoveryPress;
@InjectView(R.id.tvDiscoveryTextNormal)
TextView mTvDiscoveryTextNormal;
@InjectView(R.id.tvDiscoveryTextPress)
TextView mTvDiscoveryTextPress;
@InjectView(R.id.tvDiscoveryCount)
public TextView mTvDiscoveryCount;

@InjectView(R.id.tvMeNormal)
TextView mTvMeNormal;
@InjectView(R.id.tvMePress)
TextView mTvMePress;
@InjectView(R.id.tvMeTextNormal)
TextView mTvMeTextNormal;
@InjectView(R.id.tvMeTextPress)
TextView mTvMeTextPress;
@InjectView(R.id.tvMeCount)
public TextView mTvMeCount;


@OnClick({R.id.llMessage, R.id.llContacts, R.id.llDiscovery, R.id.llMe})
public void click(View view) {
    setTransparency();
    switch (view.getId()) {
        case R.id.llMessage:
            mVpContent.setCurrentItem(0, false);
            mTvMessagePress.getBackground().setAlpha(255);
            mTvMessageTextPress.setTextColor(Color.argb(255, 69, 192, 26));
            break;
        case R.id.llContacts:
            mVpContent.setCurrentItem(1, false);
```

```
                mTvContactsPress.getBackground().setAlpha(255);
                mTvContactsTextPress.setTextColor(Color.argb(255, 69, 192, 26));
                break;
            case R.id.llDiscovery:
                mVpContent.setCurrentItem(2, false);
                mTvDiscoveryPress.getBackground().setAlpha(255);
                mTvDiscoveryTextPress.setTextColor(Color.argb(255, 69, 192, 26));
                break;
            case R.id.llMe:
                mVpContent.setCurrentItem(3, false);
                mTvMePress.getBackground().setAlpha(255);
                mTvMeTextPress.setTextColor(Color.argb(255, 69, 192, 26));
                break;
        }
    }

    @Override
    public void init() {
        //
        registerBroadcastReceiver();
        //
        observerLineStatus();
        //
        observeUserInfoUpdate();
        //
        observeFriendChangedNotify();
        //
        observeTeamChangedNotify();
        //
        observeReceiveSystemMsg();
        //
        NIMClient.getService(MsgService.class).registerCustomAttachmentParser(new
CustomAttachParser());
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_main);
        ButterKnife.inject(this);
        initToolbar();

        //
```

```
        setTransparency();
        mTvMessagePress.getBackground().setAlpha(255);
        mTvMessageTextPress.setTextColor(Color.argb(255, 69, 192, 26));

        //ViewPager
        mVpContent.setOffscreenPageLimit(3);

    }

    @Override
    public void initData() {
        //4Fragment
        mFragments = new ArrayList<>();
        mMessageFragment = new MessageFragment();
        mContactsFragment = new ContactsFragment();
        mDiscoveryFragment = new DiscoveryFragment();
        mMeFragment = new MeFragment();
        mFragments.add(mMessageFragment);
        mFragments.add(mContactsFragment);
        mFragments.add(mDiscoveryFragment);
        mFragments.add(mMeFragment);

        //vp
        mVpContent.setAdapter(new MainPagerAdapter(getSupportFragmentManager(),
mFragments));
        mVpContent.setCurrentItem(0);

        //
        updateContactCount();
    }

    @Override
    public void initListener() {
        //vp
        mVpContent.setOnPageChangeListener(new ViewPager.OnPageChangeListener() {
            @Override
            public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
                //ViewPager
                int diaphaneity_one = (int) (255 * positionOffset);
                int diaphaneity_two = (int) (255 * (1 - positionOffset));
                switch (position) {
                    case 0:
```

```
                mTvMessageNormal.getBackground().setAlpha(diaphaneity_one);
                mTvMessagePress.getBackground().setAlpha(diaphaneity_two);
                mTvContactsNormal.getBackground().setAlpha(diaphaneity_two);
                mTvContactsPress.getBackground().setAlpha(diaphaneity_one);
                mTvMessageTextNormal.setTextColor(Color.argb(diaphaneity_one, 153, 153,
153));
                mTvMessageTextPress.setTextColor(Color.argb(diaphaneity_two, 69, 192, 26));
                mTvContactsTextNormal.setTextColor(Color.argb(diaphaneity_two, 153, 153,
153));
                mTvContactsTextPress.setTextColor(Color.argb(diaphaneity_one, 69, 192, 26));
                break;
            case 1:
                mTvContactsNormal.getBackground().setAlpha(diaphaneity_one);
                mTvContactsPress.getBackground().setAlpha(diaphaneity_two);
                mTvDiscoveryNormal.getBackground().setAlpha(diaphaneity_two);
                mTvDiscoveryPress.getBackground().setAlpha(diaphaneity_one);
                mTvContactsTextNormal.setTextColor(Color.argb(diaphaneity_one, 153, 153,
153));
                mTvContactsTextPress.setTextColor(Color.argb(diaphaneity_two, 69, 192, 26));
                mTvDiscoveryTextNormal.setTextColor(Color.argb(diaphaneity_two, 153, 153,
153));
                mTvDiscoveryTextPress.setTextColor(Color.argb(diaphaneity_one, 69, 192, 26));
                break;
            case 2:
                mTvDiscoveryNormal.getBackground().setAlpha(diaphaneity_one);
                mTvDiscoveryPress.getBackground().setAlpha(diaphaneity_two);
                mTvMeNormal.getBackground().setAlpha(diaphaneity_two);
                mTvMePress.getBackground().setAlpha(diaphaneity_one);
                mTvDiscoveryTextNormal.setTextColor(Color.argb(diaphaneity_one, 153, 153,
153));
                mTvDiscoveryTextPress.setTextColor(Color.argb(diaphaneity_two, 69, 192, 26));
                mTvMeTextNormal.setTextColor(Color.argb(diaphaneity_two, 153, 153, 153));
                mTvMeTextPress.setTextColor(Color.argb(diaphaneity_one, 69, 192, 26));
                break;
        }

    }

    @Override
    public void onPageSelected(int position) {
        //“”
        if (position == 1) {
```

```
                mContactsFragment.showQuickIndexBar(true);
            } else {
                mContactsFragment.showQuickIndexBar(false);
            }

            //positionFragment
            mFragments.get(position).initData();
        }

        @Override
        public void onPageScrollStateChanged(int state) {
            if (state != ViewPager.SCROLL_STATE_IDLE) {
                //
                mContactsFragment.showQuickIndexBar(false);
            } else {
                mContactsFragment.showQuickIndexBar(true);
            }
        }
    });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    new MenuInflater(this).inflate(R.menu.menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.itemSearch:
            Intent intent = new Intent(this, SearchUserActivity.class);
            intent.putExtra(SearchUserActivity.SEARCH_TYPE,
SearchUserActivity.SEARCH_USER_REMOTE);
            startActivity(intent);
            break;
        case R.id.itemMore:
            showMenu();
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQ_CLEAR_UNREAD) {
            updateContactCount();
        }
    }

    @Override
    protected void onDestroy() {
        unRegisterBroadcastReceiver();
        super.onDestroy();
    }

    private void initToolbar() {
        //ToolBar
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        mToolbar.setTitleTextColor(UIUtils.getColor(R.color.white));
    }

    private void showMenu() {
        View menuView = View.inflate(this, R.layout.popup_menu_main, null);
        //
        menuView.findViewById(R.id.itemCreateGroupCheat).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(MainActivity.this, TeamCheatCreateActvitiy.class));
                mPopupWindow.dismiss();
            }
        });
        //
        menuView.findViewById(R.id.itemAddFriend).setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                startActivityForResult(new Intent(MainActivity.this, NewFriendActivity.class),
MainActivity.REQ_CLEAR_UNREAD);
                mPopupWindow.dismiss();
            }
```

```
        });
        //
        menuView.findViewById(R.id.itemScan).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(MainActivity.this, ScanActivity.class));
                mPopupWindow.dismiss();
            }
        });
        //
        menuView.findViewById(R.id.itemHelpAndFeedback).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this, WebViewActivity.class);
                intent.putExtra("url", AppConst.Url.HELP_FEEDBACK);
                startActivity(intent);
                mPopupWindow.dismiss();
            }
        });
        mPopupWindow = PopupWindowFactory.getPopupWindowAtLocation(menuView,
mVpContent, Gravity.RIGHT | Gravity.TOP, UIUtils.dip2Px(12), mToolbar.getHeight() +
getStatusBarHeight());
    }

    /**
     * press()
     */
    private void setTransparency() {
        mTvMessageNormal.getBackground().setAlpha(255);
        mTvContactsNormal.getBackground().setAlpha(255);
        mTvDiscoveryNormal.getBackground().setAlpha(255);
        mTvMeNormal.getBackground().setAlpha(255);
        mTvMessagePress.getBackground().setAlpha(1);
        mTvContactsPress.getBackground().setAlpha(1);
        mTvDiscoveryPress.getBackground().setAlpha(1);
        mTvMePress.getBackground().setAlpha(1);
        mTvMessageTextNormal.setTextColor(Color.argb(255, 153, 153, 153));
        mTvContactsTextNormal.setTextColor(Color.argb(255, 153, 153, 153));
        mTvDiscoveryTextNormal.setTextColor(Color.argb(255, 153, 153, 153));
        mTvMeTextNormal.setTextColor(Color.argb(255, 153, 153, 153));
        mTvMessageTextPress.setTextColor(Color.argb(0, 69, 192, 26));
```

```
        mTvContactsTextPress.setTextColor(Color.argb(0, 69, 192, 26));
        mTvDiscoveryTextPress.setTextColor(Color.argb(0, 69, 192, 26));
        mTvMeTextPress.setTextColor(Color.argb(0, 69, 192, 26));
    }

    /**
     *
     */
    public void updateContactCount() {
        //
        List<SystemMessageType> types = new ArrayList<>();
        types.add(SystemMessageType.AddFriend);
        types.add(SystemMessageType.TeamInvite);
        int unreadCount = NimSystemSDK.querySystemMessageUnreadCountByType(types);
        if (unreadCount > 0) {
            mTvContactCount.setVisibility(View.VISIBLE);
            mTvContactCount.setText(String.valueOf(unreadCount));
            return;
        } else {
            mTvContactCount.setVisibility(View.GONE);
        }
    }

    /**
     * 22
     */
//    @Override
//    public void onBackPressed() {
//        if (exit++ == 1) {
//            App.exit();
//        } else {
//            UIUtils.showToast("");
//            new Timer().schedule(new TimerTask() {
//                @Override
//                public void run() {
//                    exit = 0;
//                }
//            }, 2000);
//        }
//    }

    /**
```

```
     *
     */
    private void registerBroadcastReceiver() {
        //
        mAuthBroadcastReceiver = new AuthBroadcastReceiver();
        registerReceiver(mAuthBroadcastReceiver, new
IntentFilter(AuthBroadcastReceiver.ACTION));
    }

    /**
     *
     */
    private void unRegisterBroadcastReceiver() {
        if (mAuthBroadcastReceiver != null) {
            unregisterReceiver(mAuthBroadcastReceiver);
            mAuthBroadcastReceiver = null;
        }
    }

    /**
     *
     */
    private void observerLineStatus() {
        mOnlineStatusObserver = new Observer<StatusCode>() {
            public void onEvent(StatusCode status) {
                LogUtils.sf("User status changed to: " + status);
                //
                if (status.wontAutoLogin()) {
                    //
                    Intent intent = new Intent();
                    intent.setAction(AuthBroadcastReceiver.ACTION);
                    intent.putExtra("status", status.getValue());
                    sendBroadcast(intent);
                }
            }
        };
        NimAccountSDK.onlineStatusListen(
                mOnlineStatusObserver, true);
    }

    /**
     *
```

```
     */
    private void observeUserInfoUpdate() {
        NimUserInfoSDK.observeUserInfoUpdate(new Observer<List<NimUserInfo>>() {
            @Override
            public void onEvent(List<NimUserInfo> nimUserInfos) {
                mMeFragment.initData();
            }
        }, true);
    }

    /**
     *
     */
    private void observeFriendChangedNotify() {
        NimFriendSDK.observeFriendChangedNotify(new Observer<FriendChangedNotify>() {
            @Override
            public void onEvent(FriendChangedNotify friendChangedNotify) {
//                List<Friend> addedOrUpdatedFriends =
friendChangedNotify.getAddedOrUpdatedFriends(); //
//                List<String> deletedFriendAccounts = friendChangedNotify.getDeletedFriends(); //

                //
                mContactsFragment.initData();
            }
        }, true);
    }

    /**
     *
     */
    private void observeTeamChangedNotify() {
        NimTeamSDK.observeTeamRemove(new Observer<Team>() {
            @Override
            public void onEvent(Team team) {
                mMessageFragment.initData();
            }
        }, true);
//        NimTeamSDK.observeTeamUpdate(new Observer<List<Team>>() {
//            @Override
//            public void onEvent(List<Team> teams) {
//                mMessageFragment.initData();
//            }
```

```
//        }, true);
    }

    /**
     *
     */
    private void observeReceiveSystemMsg() {
        NimSystemSDK.observeReceiveSystemMsg(new Observer<SystemMessage>() {
            @Override
            public void onEvent(final SystemMessage systemMessage) {

                items.clear();
                List<SystemMessageType> types = new ArrayList<>();
                types.add(SystemMessageType.AddFriend);
                types.add(SystemMessageType.TeamInvite);
                InvocationFuture<List<SystemMessage>> listInvocationFuture =
NimSystemSDK.querySystemMessageByType(types, 0, 100);
                listInvocationFuture.setCallback(new RequestCallback<List<SystemMessage>>() {
                    @Override
                    public void onSuccess(List<SystemMessage> param) {
                        if (!StringUtils.isEmpty(param)) {
                            items.addAll(param);

                            //TODO:
                            SystemMessage del = null;
                            for (SystemMessage m : items) {
                                if (m.getMessageId() != systemMessage.getMessageId() &&
                                        m.getFromAccount().equals(systemMessage.getFromAccount()) &&
m.getType() == SystemMessageType.AddFriend) {
                                    del = m;
                                    break;
                                }
                            }
                            if (del != null) {
                                items.remove(del);
                                //
                                NimSystemSDK.deleteSystemMessage(del);
                            }

                            //
                            updateContactCount();
                            mContactsFragment.updateHeaderViewUnreadCount();
```

```
                    //
                    if (systemMessage.getType() == SystemMessageType.AddFriend) {
                        NimUserInfoSDK.getUserInfoFromServer(systemMessage.getFromAccount(),
null);
                    }
                }
            }

            @Override
            public void onFailed(int code) {
            }

            @Override
            public void onException(Throwable exception) {
            }
        });
    }
}, true);
}
}
```

22:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\MsgNotificatio
nActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @ --
 */
public class MsgNotificationActivity extends BaseActivity {
```

```
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_msg_notification);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        new MenuInflater(this).inflate(R.menu.menu_one_text, menu);
        menu.getItem(0).setTitle("").setEnabled(false);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

23:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\MyCouponActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
```

```
import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @ --
 */
public class MyCouponActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_my_coupon);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

24:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\MyInfoActivity.java

```
package com.lqr.wechat.activity;
```

```
import android.content.Intent;
import android.graphics.drawable.Drawable;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.lqr.imagepicker.ImagePicker;
import com.lqr.imagepicker.bean.ImageItem;
import com.lqr.imagepicker.ui.ImageGridActivity;
import com.lqr.optionitemview.OptionItemView;
import com.lqr.wechat.R;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.UserCache;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.UIUtils;
import com.lqr.wechat.view.CustomDialog;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.ResponseCode;
import com.netease.nimlib.sdk.uinfo.constant.GenderEnum;
import com.netease.nimlib.sdk.uinfo.constant.UserInfoFieldEnum;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.io.File;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

import static com.lqr.wechat.activity.SessionActivity.IMAGE_PICKER;

/**
 * @ CSDN_LQR
```

```
 * @
 */
public class MyInfoActivity extends BaseActivity {

    Intent mIntent;
    private NimUserInfo mNimUserInfo;

    private View mGenderDialogView;
    private CustomDialog mDialog;
    private TextView mTvMale;
    private TextView mTvFemale;
    private Drawable mSelectedDrawable;
    private Drawable mUnSelectedDrawable;

    Observer<List<NimUserInfo>> userInfoUpdateObserver = new Observer<List<NimUserInfo>>()
{
        @Override
        public void onEvent(List<NimUserInfo> nimUserInfos) {
            initData();
        }
    };

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.llHeader)
    LinearLayout mLlHeader;
    @InjectView(R.id.ivHeader)
    ImageView mIvHeader;
    @InjectView(R.id.oivName)
    OptionItemView mOivName;
    @InjectView(R.id.oivQRCordCard)
    OptionItemView mOivQRCordCard;
    @InjectView(R.id.oivAccount)
    OptionItemView mOivAccount;
    @InjectView(R.id.oivGender)
    OptionItemView mOivGender;
    @InjectView(R.id.oivSignature)
    OptionItemView mOivSignature;

    @OnClick({R.id.llHeader, R.id.ivHeader, R.id.oivName, R.id.oivQRCordCard, R.id.oivGender,
R.id.oivSignature})
    public void click(View view) {
```

```
switch (view.getId()) {
    case R.id.llHeader:
        mIntent = new Intent(this, ImageGridActivity.class);
        startActivityForResult(mIntent, IMAGE_PICKER);
        break;
    case R.id.ivHeader:
        if (mNimUserInfo == null)
            return;
        mIntent = new Intent(this, ShowBigImageActivity.class);
        mIntent.putExtra("url", mNimUserInfo.getAvatar());
        startActivity(mIntent);
        break;
    case R.id.oivName:
        mIntent = new Intent(this, ChangeNameActivity.class);
        mIntent.putExtra("name", mNimUserInfo.getName());
        startActivity(mIntent);
        break;
    case R.id.oivQRCordCard:
        mIntent = new Intent(this, QRCodeCardActivity.class);
        mIntent.putExtra(QRCodeCardActivity.QRCODE_USER, mNimUserInfo);
        startActivity(mIntent);
        break;
    case R.id.oivGender:
        if (mGenderDialogView == null) {
            mGenderDialogView = View.inflate(this, R.layout.dialog_gender, null);
            mTvMale = (TextView) mGenderDialogView.findViewById(R.id.tvMale);
            mTvFemale = (TextView) mGenderDialogView.findViewById(R.id.tvFemale);
            mDialog = new CustomDialog(this, mGenderDialogView, R.style.dialog);
            mTvMale.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    updateGender(GenderEnum.MALE);
                }
            });
            mTvFemale.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    updateGender(GenderEnum.FEMALE);
                }
            });
        }
        updateGenderView(mNimUserInfo.getGenderEnum());
```

```
                mDialog.show();
                break;
            case R.id.oivSignature:
                mIntent = new Intent(this, ChangeSignatureActivity.class);
                mIntent.putExtra("signature", mNimUserInfo.getSignature());
                startActivity(mIntent);
                break;
//            case R.id.llHeader:
//                break;
        }
    }

    @Override
    public void init() {
        //
        NimUserInfoSDK.observeUserInfoUpdate(userInfoUpdateObserver, true);

        mSelectedDrawable = UIUtils.getResource().getDrawable(R.mipmap.list_selected);
        mUnSelectedDrawable = UIUtils.getResource().getDrawable(R.mipmap.list_unselected);
        mSelectedDrawable.setBounds(0, 0, mSelectedDrawable.getMinimumWidth(),
mSelectedDrawable.getMinimumHeight());
        mUnSelectedDrawable.setBounds(0, 0, mUnSelectedDrawable.getMinimumWidth(),
mUnSelectedDrawable.getMinimumHeight());
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        //
        NimUserInfoSDK.observeUserInfoUpdate(userInfoUpdateObserver, false);
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_my_info);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public void initData() {
        mNimUserInfo = NimUserInfoSDK.getUser(UserCache.getAccount());
```

```
        if (mNimUserInfo == null) {
            getUserInfoFromRemote();
        } else {
            //
            if (!TextUtils.isEmpty(mNimUserInfo.getAvatar())) {
                ImageLoaderManager.LoadNetImage(mNimUserInfo.getAvatar(), mIvHeader);
            }
            //
            mOivName.setRightText(mNimUserInfo.getName());
            mOivAccount.setRightText(mNimUserInfo.getAccount());
            mOivSignature.setRightText(TextUtils.isEmpty(mNimUserInfo.getSignature()) ? "" :
mNimUserInfo.getSignature());
            mOivGender.setRightText(mNimUserInfo.getGenderEnum() == GenderEnum.FEMALE ? "
" : mNimUserInfo.getGenderEnum() == GenderEnum.MALE ? "" : "");
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == ImagePicker.RESULT_CODE_ITEMS) {//
            if (data != null) {
                //
//                boolean isOrig = data.getBooleanExtra(ImagePreviewActivity.ISORIGIN, false);
                showWaitingDialog("...");
                ArrayList<ImageItem> images = (ArrayList<ImageItem>)
data.getSerializableExtra(ImagePicker.EXTRA_RESULT_ITEMS);
                if (images != null && images.size() > 0) {
                    //
                    File file = new File(images.get(0).path);
                    NimUserInfoSDK.uploadFile(file, "image/jpeg", new
RequestCallbackWrapper<String>() {
```

```
                @Override
                public void onResult(int code, String url, Throwable exception) {

                    if (code == ResponseCode.RES_SUCCESS
                            && !TextUtils.isEmpty(url)) {// Url
                        Map<UserInfoFieldEnum, Object> fields = new HashMap<UserInfoFieldEnum,
Object>(
                                1);
                        fields.put(UserInfoFieldEnum.AVATAR, url);
                    }

                    Map<UserInfoFieldEnum, Object> fields = new HashMap(1);
                    fields.put(UserInfoFieldEnum.AVATAR, url);
                    NimUserInfoSDK.updateUserInfo(fields, new RequestCallbackWrapper<Void>()
{
                        @Override
                        public void onResult(int code, Void result, Throwable exception) {
                            if (code == ResponseCode.RES_SUCCESS) {//
                                UIUtils.showToast("");
                                getUserInfoFromRemote();//
                            } else {//
                                UIUtils.showToast("");
                            }
                            hideWaitingDialog();
                        }
                    });
                }
            });
        }
    }
}

private void initToolbar() {
    setSupportActionBar(mToolbar);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setTitle("");
    mToolbar.setNavigationIcon(R.mipmap.ic_back);
}

private void getUserInfoFromRemote() {
    List<String> accountList = new ArrayList<>();
```

```
        accountList.add(UserCache.getAccount());
        NimUserInfoSDK.getUserInfosFormServer(accountList, new
RequestCallback<List<NimUserInfo>>() {
            @Override
            public void onSuccess(List<NimUserInfo> param) {
                initData();
            }

            @Override
            public void onFailed(int code) {
                UIUtils.showToast("" + code);
            }

            @Override
            public void onException(Throwable exception) {
                exception.printStackTrace();
            }
        });
    }

    private void updateGender(final GenderEnum gender) {
        updateGenderView(gender);
        showWaitingDialog("");
        Map<UserInfoFieldEnum, Object> fields = new HashMap(1);
        fields.put(UserInfoFieldEnum.GENDER, gender.getValue());
        NimUserInfoSDK.updateUserInfo(fields, new RequestCallbackWrapper<Void>() {
            @Override
            public void onResult(int code, Void result, Throwable exception) {
                hideWaitingDialog();
                if (code == ResponseCode.RES_SUCCESS) {
                    UIUtils.showToast("");
                    mDialog.dismiss();
                } else {
                    UIUtils.showToast("");
                }
            }
        });
    }

    private void updateGenderView(GenderEnum gender) {
        if (gender == GenderEnum.MALE) {
            mTvMale.setCompoundDrawables(null, null, mSelectedDrawable, null);
```

```
            mTvFemale.setCompoundDrawables(null, null, mUnSelectedDrawable, null);
        } else if (gender == GenderEnum.FEMALE) {
            mTvMale.setCompoundDrawables(null, null, mUnSelectedDrawable, null);
            mTvFemale.setCompoundDrawables(null, null, mSelectedDrawable, null);
        } else {
            mTvMale.setCompoundDrawables(null, null, mUnSelectedDrawable, null);
            mTvFemale.setCompoundDrawables(null, null, mUnSelectedDrawable, null);
        }
    }
}
```

25:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\NearbyPerpleActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */

public class NearbyPerpleActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_nearby_perple);
        ButterKnife.inject(this);
        initToolbar();
        showMaterialDialog("", "", "", "", new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                hideMaterialDialog();
```

```
            }
        }, new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                hideMaterialDialog();
            }
        });
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

26:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\NewFriendActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
```

```
import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
import com.lqr.recyclerview.LQRRecyclerView;
import com.lqr.wechat.R;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.NewFriend;
import com.lqr.wechat.nimsdk.NimFriendSDK;
import com.lqr.wechat.nimsdk.NimSystemSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.StringUtils;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.InvocationFuture;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.msg.constant.SystemMessageType;
import com.netease.nimlib.sdk.msg.model.SystemMessage;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class NewFriendActivity extends BaseActivity {

    private Intent mIntent;
    private List<NewFriend> mNewFriendList = new ArrayList<>();
    private LQRAdapterForRecyclerView<NewFriend> mAdapter;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.etContent)
    EditText mEtContent;
    @InjectView(R.id.tvNewFriend)
    TextView mTvNewFriend;
    @InjectView(R.id.rvNewFriend)
    LQRRecyclerView mRvNewFriend;
```

```
    @OnClick({R.id.etContent})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.etContent:
                mIntent = new Intent(this, SearchUserActivity.class);
                mIntent.putExtra(SearchUserActivity.SEARCH_TYPE,
SearchUserActivity.SEARCH_USER_LOCAL);
                startActivity(mIntent);
                break;
        }
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_new_friend);
        ButterKnife.inject(this);
        initToolbar();

        //
        List<SystemMessageType> types = new ArrayList<>();
        types.add(SystemMessageType.AddFriend);
        NimSystemSDK.resetSystemMessageUnreadCount(types);
    }

    @Override
    public void initData() {
        showWaitingDialog("");
        //1
        List<SystemMessageType> types = new ArrayList<>();
        types.add(SystemMessageType.AddFriend);
        InvocationFuture<List<SystemMessage>> listInvocationFuture =
NimSystemSDK.querySystemMessageByType(types, 0, 100);
        listInvocationFuture.setCallback(new RequestCallback<List<SystemMessage>>() {
            @Override
            public void onSuccess(final List<SystemMessage> smList) {
                //2
                List<String> accounts = new ArrayList<>();
                for (SystemMessage msg : smList) {
                    accounts.add(msg.getFromAccount());
                }
                if (StringUtils.isEmpty(accounts)) {
```

```
                mTvNewFriend.setVisibility(View.GONE);
                loadDone();
                return;
            } else {
                mTvNewFriend.setVisibility(View.VISIBLE);
                NimUserInfoSDK.getUserInfosFormServer(accounts, new
RequestCallback<List<NimUserInfo>>() {
                    @Override
                    public void onSuccess(List<NimUserInfo> userInfoList) {
                        mNewFriendList.clear();
                        //3
                        for (int i = 0; i < userInfoList.size(); i++) {
                            NimUserInfo userInfo = userInfoList.get(i);
                            mNewFriendList.add(new NewFriend(userInfo, smList.get(i).getContent()));
                        }
                        setAdapter();
                        loadDone();
                    }

                    @Override
                    public void onFailed(int code) {
                        loadDone();
                    }

                    @Override
                    public void onException(Throwable exception) {
                        loadDone();
                    }
                });
            }

        }

        @Override
        public void onFailed(int code) {
            UIUtils.showToast("" + code);
            loadDone();
        }

        @Override
        public void onException(Throwable exception) {
            exception.printStackTrace();
```

```
            loadDone();
        }
    });
}

private void loadDone() {
    UIUtils.postTaskSafely(new Runnable() {
        @Override
        public void run() {
            hideWaitingDialog();
        }
    });
}

private void setAdapter() {
    if (mAdapter == null) {
        mAdapter = new LQRAdapterForRecyclerView<NewFriend>(this,
R.layout.item_new_friends_rv, mNewFriendList) {
            @Override
            public void convert(LQRViewHolderForRecyclerView helper, final NewFriend item, final
int position) {
                if (!TextUtils.isEmpty(item.getUserInfo().getAvatar())) {
                    ImageLoaderManager.LoadNetImage(item.getUserInfo().getAvatar(), (ImageView)
helper.getView(R.id.ivHeader));
                } else {
                    ((ImageView)
helper.getView(R.id.ivHeader)).setImageResource(R.mipmap.default_header);
                }
                helper.setText(R.id.tvName, item.getUserInfo().getName()).setText(R.id.tvMsg,
TextUtils.isEmpty(item.getMsg()) ? "" : item.getMsg());

                if (NimFriendSDK.isMyFriend(item.getUserInfo().getAccount())) {
                    helper.setViewVisibility(R.id.tvAdded, View.VISIBLE)
                            .setViewVisibility(R.id.btnAck, View.GONE);
                } else {
                    helper.setViewVisibility(R.id.tvAdded, View.GONE)
                            .setViewVisibility(R.id.btnAck, View.VISIBLE);
                }

                helper.getView(R.id.btnAck).setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
```

```
                    NimFriendSDK.ackAddFriendRequest(item.getUserInfo().getAccount(), true);
                    UIUtils.postTaskDelay(new Runnable() {
                        @Override
                        public void run() {
                            mAdapter.notifyItemChanged(position);
                        }
                    }, 500);

                }
            });
        }
    };
    mRvNewFriend.setAdapter(mAdapter);
  } else {
    mAdapter.notifyDataSetChanged();
  }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
  new MenuInflater(this).inflate(R.menu.menu_one_text, menu);
  menu.getItem(0).setTitle("");
  return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case android.R.id.home:
      finish();
      break;
    case R.id.itemOne:
      startActivity(new Intent(this, AddFriendActivity.class));
      break;
  }
  return super.onOptionsItemSelected(item);
}

private void initToolbar() {
  setSupportActionBar(mToolbar);
  getSupportActionBar().setTitle("");
  getSupportActionBar().setDisplayHomeAsUpEnabled(true);
```

```
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }


}
```

27:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\NewMsgNotifySetActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class NewMsgNotifySetActivity extends BaseActivity {

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_new_msg_notify_set);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
```

```
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

28:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\OtherLoginActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.lqr.wechat.R;
import com.lqr.wechat.utils.UIUtils;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class OtherLoginActivity extends BaseActivity {

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @InjectView(R.id.etPhone)
    EditText mEtPhone;
    @InjectView(R.id.etPwd)
    EditText mEtPwd;
```

```
@InjectView(R.id.vLinePhone)
View mVLinePhone;
@InjectView(R.id.vLinePwd)
View mVLinePwd;

@InjectView(R.id.btnLogin)
Button mBtnLogin;

@Override
public void initView() {
    setContentView(R.layout.activity_other_login);
    ButterKnife.inject(this);

    initToolbar();
}

@Override
public void initListener() {
    /*------------------  begin ------------------*/
    mEtPhone.setOnFocusChangeListener(new View.OnFocusChangeListener() {
        @Override
        public void onFocusChange(View v, boolean hasFocus) {
            if (hasFocus) {
                mVLinePhone.setBackgroundColor(UIUtils.getColor(R.color.green0));
            } else {
                mVLinePhone.setBackgroundColor(UIUtils.getColor(R.color.line));
            }
        }
    });
    mEtPwd.setOnFocusChangeListener(new View.OnFocusChangeListener() {
        @Override
        public void onFocusChange(View v, boolean hasFocus) {
            if (hasFocus) {
                mVLinePwd.setBackgroundColor(UIUtils.getColor(R.color.green0));
            } else {
                mVLinePwd.setBackgroundColor(UIUtils.getColor(R.color.line));
            }
        }
    });
    mEtPhone.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {
```

```
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            if (!TextUtils.isEmpty(mEtPhone.getText().toString()) &&
!TextUtils.isEmpty(mEtPwd.getText().toString())) {
                mBtnLogin.setEnabled(true);
            } else {
                mBtnLogin.setEnabled(false);
            }
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    });
    mEtPwd.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            if (!TextUtils.isEmpty(mEtPhone.getText().toString()) &&
!TextUtils.isEmpty(mEtPwd.getText().toString())) {
                mBtnLogin.setEnabled(true);
            } else {
                mBtnLogin.setEnabled(false);
            }
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    });
    /*------------------  end ------------------*/
    super.initListener();
}
```

```
    /**
     * Toolbar
     */
    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

29:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\PhotoActivity.java

```
package com.lqr.wechat.activity;

import android.app.Activity;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.view.View;

import com.lqr.wechat.R;
import com.lqr.wechat.utils.Bimp;
import com.lqr.wechat.utils.SDCardUtils;
import com.lqr.wechat.utils.StringUtils;
```

```
import com.lqr.wechat.view.ZoomImageView;
import com.nostra13.universalimageloader.core.ImageLoader;

import java.io.File;
import java.util.UUID;

/**
 * @ CSDN_LQR
 * @ Activity
 * <p>
 *
 * Intent intent = new Intent(getActivity(), PhotoActivity.class);
 * intent.putExtra("flag", 2);//12
 * intent.putExtra("noZoom", 1);//
 * intent.putExtra("noSurePic", 1);//
 * getActivity().startActivityForResult(intent, MainActivity.SELECT_BAR_CODE_PHOTO);
 */
public class PhotoActivity extends Activity {

    private View cancelBtn;
    private View sendBtn;
    private ZoomImageView img;

    private static final int TAKE_PHOTO = 110;
    private static final int SELECT_PHOTO = 111;
    private static final int CUT_PHOTO_REQUEST_CODE = 112;

    int flag = 0;
    int noZoom = 0;//0zoom1zoom
    int noSurePic = 0;//01

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_photo);

        initView();

        flag = getIntent().getIntExtra("flag", 0);
        noZoom = getIntent().getIntExtra("noZoom", 0);
        noSurePic = getIntent().getIntExtra("noSurePic", 0);
```

```
    if (flag == 0) {
        String imgUrl = getIntent().getStringExtra("imgUrl");
        if (!StringUtils.isEmpty(imgUrl)) {
            ImageLoader.getInstance().displayImage(imgUrl, img);
        } else {
            this.finish();
        }
    } else if (flag == 1) {
        photo();
    } else if (flag == 2) {
        Intent intent = new Intent(
                Intent.ACTION_PICK,
                MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(intent, SELECT_PHOTO);
    }

}

private void initView() {
    cancelBtn = findViewById(R.id.photo_cancel);
    sendBtn = findViewById(R.id.photo_send);
    img = (ZoomImageView) findViewById(R.id.photo_img);
    cancelBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            PhotoActivity.this.finish();
        }
    });
    sendBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            sendPic();
        }
    });
}

/**
 *
 */
private void sendPic() {
    Bundle b = new Bundle();
    b.putString("imgPath", filePath);
```

```
    Intent result = new Intent();
    result.putExtras(b);
    setResult(Activity.RESULT_OK, result);
    finish();
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && (null != data || requestCode == TAKE_PHOTO)) {
        switch (requestCode) {//
            case TAKE_PHOTO:
                if (photoUri != null) {
                    if (noZoom == 0) {
                        startPhotoZoom(photoUri);
                    } else {
                        filePath = getRealFilePath(PhotoActivity.this, photoUri);
                        if (noSurePic == 0) {
                            Bitmap bitmap = Bimp.zoomForFilePath(PhotoActivity.this, filePath);
                            img.setImageBitmap(bitmap);
                        } else {
                            sendPic();
                        }

                    }
                } else {
                    PhotoActivity.this.finish();
                }
                break;
            case SELECT_PHOTO://
                Uri uri = data.getData();
                if (uri != null) {
                    if (noZoom == 0) {
                        startPhotoZoom(uri);
                    } else {
                        filePath = getRealFilePath(PhotoActivity.this, uri);
                        if (noSurePic == 0) {
                            Bitmap bitmap = Bimp.zoomForFilePath(PhotoActivity.this, filePath);
                            img.setImageBitmap(bitmap);
                        } else {
                            sendPic();
                        }
```

```
                }
            } else {
                PhotoActivity.this.finish();
            }
            break;
        case CUT_PHOTO_REQUEST_CODE://
            Bitmap bitmap = Bimp.zoomForFilePath(PhotoActivity.this, filePath);
            img.setImageBitmap(bitmap);
            break;
        }
    } else {
        PhotoActivity.this.finish();
    }
}

private String filePath;

private void startPhotoZoom(Uri uri) {
    try {
        String address = UUID.randomUUID() + "";
        File destDir = new File(SDCardUtils.getSDCardPath() + "/CSDN_LQR/img");
        if (!destDir.exists()) {
            destDir.mkdirs();
        }
//          Toast.makeText(PhotoActivity.this, uri.getPath(), Toast.LENGTH_LONG).show();
        Uri imageUri = Uri.parse("file:///sdcard/CSDN_LQR/img/" + address + ".jpg");
        filePath = imageUri.getPath();

        final Intent intent = new Intent("com.android.camera.action.CROP");

        // URL
        intent.setDataAndType(uri, "image/*");

        intent.putExtra("crop", "true");
        intent.putExtra("aspectX", 1);
        intent.putExtra("aspectY", 1);
        intent.putExtra("outputX", 720);
        intent.putExtra("outputY", 720);
        //
        intent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
        //
        intent.putExtra("outputFormat",
```

```
                Bitmap.CompressFormat.JPEG.toString());
        //
        intent.putExtra("noFaceDetection", false);
        intent.putExtra("return-data", false);
        startActivityForResult(intent, CUT_PHOTO_REQUEST_CODE);

      } catch (Exception e) {
        e.printStackTrace();
      }
    }

    private String path;
    private Uri photoUri;

    public void photo() {
      Intent openCameraIntent = new Intent(
          MediaStore.ACTION_IMAGE_CAPTURE);

      String sdcardState = Environment.getExternalStorageState();
      String sdcardPathDir = Environment
          .getExternalStorageDirectory().getPath() + "/CSDN_LQR/img/";
      File file = null;
      if (Environment.MEDIA_MOUNTED.equals(sdcardState)) {
        File fileDir = new File(sdcardPathDir);
        if (!fileDir.exists()) {
          fileDir.mkdirs();
        }
        file = new File(sdcardPathDir + System.currentTimeMillis() + ".jpg");
      }
      if (file != null) {
        path = file.getPath();
//          photoUri = Uri.fromFile(file);
        photoUri = Uri.parse("file://"+file.getAbsolutePath());
        openCameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoUri);

        startActivityForResult(openCameraIntent, TAKE_PHOTO);
      }
    }

    /**
     *
     *
```

```
     * @param context
     * @param uri
     * @return
     */
    public static String getRealFilePath(final Context context, final Uri uri) {
        if (null == uri) return null;
        final String scheme = uri.getScheme();
        String data = null;
        if (scheme == null)
            data = uri.getPath();
        else if (ContentResolver.SCHEME_FILE.equals(scheme)) {
            data = uri.getPath();
        } else if (ContentResolver.SCHEME_CONTENT.equals(scheme)) {
            Cursor cursor = context.getContentResolver().query(uri, new
String[]{MediaStore.Images.ImageColumns.DATA}, null, null, null);
            if (null != cursor) {
                if (cursor.moveToFirst()) {
                    int index = cursor.getColumnIndex(MediaStore.Images.ImageColumns.DATA);
                    if (index > -1) {
                        data = cursor.getString(index);
                    }
                }
                cursor.close();
            }
        }
        return data;
    }

}
```

30:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\PostscriptActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimFriendSDK;
```

```
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.RequestCallback;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class PostscriptActivity extends BaseActivity {

    public String mAccount;//
    public String mMsg;//

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.btnOk)
    Button mBtnOk;

    @InjectView(R.id.etMsg)
    EditText mEtMsg;

    @OnClick({R.id.btnOk, R.id.ibClear})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.btnOk:
                showWaitingDialog("");
                mMsg = mEtMsg.getText().toString();
                //
                NimFriendSDK.addFriend(mAccount, mMsg, new RequestCallback<Void>() {
                    @Override
                    public void onSuccess(Void param) {
                        hideWaitingDialog();
                        UIUtils.showToast("");
                        finish();
                    }

                    @Override
                    public void onFailed(int code) {
                        UIUtils.showToast("" + code);
```

```
                hideWaitingDialog();
            }

            @Override
            public void onException(Throwable exception) {
                exception.printStackTrace();
                hideWaitingDialog();
            }
        });
        break;
    case R.id.ibClear:
        mEtMsg.setText("");
        break;
  }
}

@Override
public void init() {
  mAccount = getIntent().getStringExtra("account");
}

@Override
public void initView() {
  setContentView(R.layout.activity_postscript);
  ButterKnife.inject(this);
  initToolbar();
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case android.R.id.home:
      finish();
      break;
  }
  return super.onOptionsItemSelected(item);
}

private void initToolbar() {
  mToolbar.setNavigationIcon(R.mipmap.ic_back);
  setSupportActionBar(mToolbar);
  getSupportActionBar().setTitle(UIUtils.getString(R.string.app_name));
```

```
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mBtnOk.setVisibility(View.VISIBLE);
        mBtnOk.setText("");
    }
}
```

31:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\PrivacySetActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class PrivacySetActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_privacy_set);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }
```

```
    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

32:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\QRCodeCardActivity.java

```
package com.lqr.wechat.activity;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import com.lqr.ninegridimageview.LQRNineGridImageView;
import com.lqr.ninegridimageview.LQRNineGridImageViewAdapter;
import com.lqr.wechat.AppConst;
import com.lqr.wechat.R;
import com.lqr.wechat.factory.ThreadPoolFactory;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.team.model.Team;
import com.netease.nimlib.sdk.team.model.TeamMember;
import com.netease.nimlib.sdk.uinfo.constant.GenderEnum;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.ArrayList;
import java.util.List;
```

```
import butterknife.ButterKnife;
import butterknife.InjectView;
import cn.bingoogolapple.qrcode.zxing.QRCodeEncoder;

/**
 * @ CSDN_LQR
 * @
 */
public class QRCodeCardActivity extends BaseActivity {

    public static final String QRCODE_USER = "code_user";
    public static final String QRCODE_TEAM = "code_team";

    private NimUserInfo mUser;
    private Team mTeam;
    private boolean isUserInfoQRcode = true;
    private LQRNineGridImageViewAdapter<NimUserInfo> mNineGridAdapter;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.ivHeader)
    ImageView mIvHeader;
    @InjectView(R.id.ngiv)
    LQRNineGridImageView mNgivHeader;
    @InjectView(R.id.tvName)
    TextView mTtvName;
    @InjectView(R.id.ivGender)
    ImageView mIvGender;
    @InjectView(R.id.ivCard)
    ImageView mIvCard;
    @InjectView(R.id.tvTip)
    TextView mTvTip;

    @Override
    public void init() {
        Intent intent = getIntent();
        mUser = (NimUserInfo) intent.getSerializableExtra(QRCODE_USER);
        mTeam = (Team) intent.getSerializableExtra(QRCODE_TEAM);
        if (mUser == null && mTeam == null)
            interrupt();
        if (mUser != null && mTeam == null) {
            isUserInfoQRcode = true;
```

```
        } else {
            isUserInfoQRcode = false;
        }
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_qrcode_card);
        ButterKnife.inject(this);
        initToolbar();
        mNineGridAdapter = new LQRNineGridImageViewAdapter<NimUserInfo>() {
            @Override
            protected void onDisplayImage(Context context, ImageView imageView, NimUserInfo
userInfo) {
                if (!TextUtils.isEmpty(userInfo.getAvatar())) {
                    ImageLoaderManager.LoadNetImage(userInfo.getAvatar(), imageView);
                } else {
                    imageView.setImageResource(R.mipmap.default_header);
                }
            }
        };

        if (isUserInfoQRcode) {
            mIvHeader.setVisibility(View.VISIBLE);
            mNgivHeader.setVisibility(View.GONE);
            final String avatar = mUser.getAvatar();

            if (!TextUtils.isEmpty(avatar)) {
                ImageLoaderManager.LoadNetImage(avatar, mIvHeader);
            } else {
                mIvHeader.setImageResource(R.mipmap.default_header);
            }
            mTtvName.setText(mUser.getName());
            mTvTip.setText("");
            if (mUser.getGenderEnum() == GenderEnum.FEMALE) {
                mIvGender.setImageResource(R.mipmap.ic_gender_female);
            } else if (mUser.getGenderEnum() == GenderEnum.MALE) {
                mIvGender.setImageResource(R.mipmap.ic_gender_male);
            } else {
                mIvGender.setVisibility(View.GONE);
            }
        } else {
```

```
        mIvHeader.setVisibility(View.GONE);
        mNgivHeader.setVisibility(View.VISIBLE);
        mTtvName.setText(TextUtils.isEmpty(mTeam.getName()) ? "(" +
mTeam.getMemberCount() + ")" : mTeam.getName());
        mIvGender.setVisibility(View.GONE);
        mTvTip.setText("");
        //
        NimTeamSDK.queryMemberList(mTeam.getId(), new
RequestCallback<List<TeamMember>>() {
            @Override
            public void onSuccess(List<TeamMember> memberList) {
                if (memberList != null && memberList.size() > 0) {
                    List<String> accounts = new ArrayList<>();
                    int count = memberList.size() > 9 ? 9 : memberList.size();
                    for (int i = 0; i < count; i++) {
                        accounts.add(memberList.get(i).getAccount());
                    }
                    NimUserInfoSDK.getUserInfosFormServer(accounts, new
RequestCallback<List<NimUserInfo>>() {
                        @Override
                        public void onSuccess(List<NimUserInfo> result) {
                            mNgivHeader.setAdapter(mNineGridAdapter);
                            mNgivHeader.setImagesData(result);
                        }

                        @Override
                        public void onFailed(int code) {

                        }

                        @Override
                        public void onException(Throwable exception) {

                        }
                    });
                }
            }

            @Override
            public void onFailed(int code) {

            }
```

```
            @Override
            public void onException(Throwable exception) {

            }
        });
    }
    createQRCode();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
//      new MenuInflater(this).inflate(R.menu.menu_more, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            break;
        case R.id.itemMore:
            break;
    }
    return super.onOptionsItemSelected(item);
}

private void initToolbar() {
    setSupportActionBar(mToolbar);
    getSupportActionBar().setTitle(isUserInfoQRcode ? "" : "");
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    mToolbar.setNavigationIcon(R.mipmap.ic_back);
}

private void createQRCode() {
    ThreadPoolFactory.getNormalPool().execute(new Runnable() {
        @Override
        public void run() {
            String content = isUserInfoQRcode ? AppConst.QRCodeCommend.ACCOUNT +
mUser.getAccount() : AppConst.QRCodeCommend.TEAMID + mTeam.getId();
            final Bitmap codeWithLogo5 = QRCodeEncoder.syncEncodeQRCode(content,
```

```
UIUtils.dip2Px(200));
                UIUtils.postTaskSafely(new Runnable() {
                    @Override
                    public void run() {
                        mIvCard.setImageBitmap(codeWithLogo5);
                    }
                });
            }
        });
    }
}
```

33:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\RedPacketActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.lqr.wechat.R;
import com.lqr.wechat.utils.UIUtils;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class RedPacketActivity extends BaseActivity {

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
```

```
@InjectView(R.id.tvTip)
TextView mTvTip;

@InjectView(R.id.tvMoneyLeft)
TextView mTvMoneyLeft;
@InjectView(R.id.etMoney)
EditText mEtMoney;
@InjectView(R.id.tvMoneyRight)
TextView mTvMoneyRight;

@InjectView(R.id.etMessage)
EditText mEtMessage;
@InjectView(R.id.tvHint)
TextView mTvHint;

@InjectView(R.id.tvMoney)
TextView mTvMoney;
@InjectView(R.id.btnOk)
Button mBtnOk;

@Override
public void initView() {
    setContentView(R.layout.activity_red_packet);
    ButterKnife.inject(this);
    initToolbar();
}

@Override
public void initListener() {
    mEtMoney.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            if (TextUtils.isEmpty(s)) {
                mBtnOk.setEnabled(false);
                mTvMoney.setText("0.00");
            } else {
```

```
            mBtnOk.setEnabled(true);
            String result = String.format("%.2f", Double.valueOf(s.toString()));
            mTvMoney.setText("" + result);
            Double money = Double.valueOf(result);
            if (money > 200) {
                mTvTip.setVisibility(View.VISIBLE);
                mTvMoneyLeft.setTextColor(UIUtils.getColor(R.color.red5));
                mEtMoney.setTextColor(UIUtils.getColor(R.color.red5));
                mTvMoneyRight.setTextColor(UIUtils.getColor(R.color.red5));
                mBtnOk.setEnabled(false);
            } else {
                mTvTip.setVisibility(View.GONE);
                mTvMoneyLeft.setTextColor(UIUtils.getColor(R.color.black0));
                mEtMoney.setTextColor(UIUtils.getColor(R.color.black0));
                mTvMoneyRight.setTextColor(UIUtils.getColor(R.color.black0));
                mBtnOk.setEnabled(true);
            }
        }
    }

    @Override
    public void afterTextChanged(Editable s) {

    }
});
mEtMessage.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        mTvHint.setVisibility(TextUtils.isEmpty(s) ? View.VISIBLE : View.GONE);
        if (s.length() > 25) {
            mEtMessage.setText(s.subSequence(0, 25));
            mEtMessage.setSelection(25);
        }
    }

    @Override
    public void afterTextChanged(Editable s) {
```

```
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        new MenuInflater(this).inflate(R.menu.menu_one_icon, menu);
        menu.getItem(0).setIcon(R.mipmap.ic_question);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
            case R.id.itemOne:
                UIUtils.showToast("");
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setSubtitle("");
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
        mToolbar.setBackgroundColor(UIUtils.getColor(R.color.red1));
    }

}
```

34:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\ScanActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.os.Vibrator;
```

```
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.PopupWindow;
import android.widget.TextView;

import com.lqr.imagepicker.ImagePicker;
import com.lqr.imagepicker.bean.ImageItem;
import com.lqr.imagepicker.ui.ImageGridActivity;
import com.lqr.wechat.AppConst;
import com.lqr.wechat.R;
import com.lqr.wechat.factory.PopupWindowFactory;
import com.lqr.wechat.factory.ThreadPoolFactory;
import com.lqr.wechat.model.UserCache;
import com.lqr.wechat.nimsdk.NimFriendSDK;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.utils.LogUtils;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.team.model.Team;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;
import cn.bingoogolapple.qrcode.core.QRCodeView;
import cn.bingoogolapple.qrcode.zxing.QRCodeDecoder;
import cn.bingoogolapple.qrcode.zxing.ZXingView;

import static com.lqr.wechat.activity.SessionActivity.IMAGE_PICKER;
```

```
/**
 * @ CSDN_LQR
 * @
 */
public class ScanActivity extends BaseActivity implements QRCodeView.Delegate {

    private FrameLayout mView;
    private PopupWindow mPopupWindow;

    @InjectView(R.id.zxingview)
    ZXingView mZxingview;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @InjectView(R.id.llSaoma)
    LinearLayout mLlSaoma;
    @InjectView(R.id.llFengmian)
    LinearLayout mLlFengmian;
    @InjectView(R.id.llJiejing)
    LinearLayout mLlJiejing;
    @InjectView(R.id.llFanyi)
    LinearLayout mLlFanyi;

    @InjectView(R.id.ivSaomaPress)
    ImageView mIvSaomaPress;
    @InjectView(R.id.ivSaomaNormal)
    ImageView mIvSaomaNormal;
    @InjectView(R.id.ivFengmianPress)
    ImageView mIvFengmianPress;
    @InjectView(R.id.ivFengmianNormal)
    ImageView mIvFengmianNormal;
    @InjectView(R.id.ivJiejingPress)
    ImageView mIvJiejingPress;
    @InjectView(R.id.ivJiejingNormal)
    ImageView mIvJiejingNormal;
    @InjectView(R.id.ivFanyiPress)
    ImageView mIvFanyiPress;
    @InjectView(R.id.ivFanyiNormal)
    ImageView mIvFanyiNormal;

    @OnClick({R.id.llSaoma, R.id.llFengmian, R.id.llJiejing, R.id.llFanyi})
```

```
public void click(View view) {
    switch (view.getId()) {
        case R.id.llSaoma:
            selectBottomOne(0);
            break;
        case R.id.llFengmian:
            selectBottomOne(1);
            break;
        case R.id.llJiejing:
            selectBottomOne(2);
            break;
        case R.id.llFanyi:
            selectBottomOne(3);
            break;
    }
}

@Override
public void initView() {
    setContentView(R.layout.activity_scan);
    ButterKnife.inject(this);
    initToolbar();
    selectBottomOne(0);
}

@Override
public void initListener() {
    mZxingview.setDelegate(this);
}

@Override
protected void onStart() {
    super.onStart();
    mZxingview.startCamera();
    mZxingview.startSpotAndShowRect();
}

@Override
protected void onStop() {
    super.onStop();
    mZxingview.stopCamera();
}
```

```
@Override
protected void onDestroy() {
    super.onDestroy();
    mZxingview.onDestroy();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    new MenuInflater(this).inflate(R.menu.menu_more, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            break;
        case R.id.itemMore:
            showPopupMenu();
            break;
    }
    return super.onOptionsItemSelected(item);
}

private void initToolbar() {
    setSupportActionBar(mToolbar);
    getSupportActionBar().setTitle("/");
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    mToolbar.setNavigationIcon(R.mipmap.ic_back);
}

public void selectBottomOne(int switchItem) {
    mIvSaomaPress.setVisibility(View.GONE);
    mIvFengmianPress.setVisibility(View.GONE);
    mIvJiejingPress.setVisibility(View.GONE);
    mIvFanyiPress.setVisibility(View.GONE);
    switch (switchItem) {
        case 0:
            getSupportActionBar().setTitle("/");
            mIvSaomaPress.setVisibility(View.VISIBLE);
```

```
            break;
        case 1:
            getSupportActionBar().setTitle("/");
            mIvFengmianPress.setVisibility(View.VISIBLE);
            break;
        case 2:
            getSupportActionBar().setTitle("");
            mIvJiejingPress.setVisibility(View.VISIBLE);
            break;
        case 3:
            getSupportActionBar().setTitle("");
            mIvFanyiPress.setVisibility(View.VISIBLE);
            break;
    }
  }

  private void showPopupMenu() {
    if (mView == null) {
        mView = new FrameLayout(this);
        mView.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT));
        mView.setBackgroundColor(UIUtils.getColor(R.color.white));

        TextView tv = new TextView(this);
        FrameLayout.LayoutParams params = new
FrameLayout.LayoutParams(FrameLayout.LayoutParams.MATCH_PARENT, UIUtils.dip2Px(45));
        tv.setLayoutParams(params);
        tv.setGravity(Gravity.LEFT | Gravity.CENTER_VERTICAL);
        tv.setPadding(UIUtils.dip2Px(20), 0, 0, 0);
        tv.setTextColor(UIUtils.getColor(R.color.gray0));
        tv.setTextSize(14);
        tv.setText("");
        mView.addView(tv);

        tv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mPopupWindow.dismiss();
                Intent intent = new Intent(ScanActivity.this, ImageGridActivity.class);
                startActivityForResult(intent, IMAGE_PICKER);
            }
```

```
            });
        }
        mPopupWindow = PopupWindowFactory.getPopupWindowAtLocation(mView,
ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT,
getWindow().getDecorView().getRootView(), Gravity.BOTTOM, 0, 0);
        mPopupWindow.setOnDismissListener(new PopupWindow.OnDismissListener() {
            @Override
            public void onDismiss() {
                PopupWindowFactory.makeWindowLight(ScanActivity.this);
            }
        });
        PopupWindowFactory.makeWindowDark(this);
    }

    private void vibrate() {
        Vibrator vibrator = (Vibrator) getSystemService(VIBRATOR_SERVICE);
        vibrator.vibrate(200);
    }

    @Override
    public void onScanQRCodeSuccess(String result) {
        LogUtils.sf(result);
        handleResult(result);
    }

    private void handleResult(String result) {
        vibrate();
        mZxingview.startSpot();
        //
        if (result.startsWith(AppConst.QRCodeCommend.ACCOUNT)) {
            String account = result.substring(AppConst.QRCodeCommend.ACCOUNT.length());
//            UIUtils.showToast("" + account);
            if (NimFriendSDK.isMyFriend(account)) {
                UIUtils.showToast("");
                return;
            }
            Intent intent = new Intent(ScanActivity.this, PostscriptActivity.class);
            intent.putExtra("account", account);
            startActivity(intent);
        }
        //
        else if (result.startsWith(AppConst.QRCodeCommend.TEAMID)) {
```

```
final String teamId = result.substring(AppConst.QRCodeCommend.TEAMID.length());
NimTeamSDK.searchTeam(teamId, new RequestCallback<Team>() {
    @Override
    public void onSuccess(Team team) {
        if (team.isMyTeam()) {
            UIUtils.showToast("");
        } else {
            List<String> accounts = new ArrayList<String>(1);
            accounts.add(UserCache.getAccount());
            NimTeamSDK.addMembers(teamId, accounts, new RequestCallback<Void>() {
                @Override
                public void onSuccess(Void param) {
                    //
                    Intent intent = new Intent(ScanActivity.this, SessionActivity.class);
                    intent.putExtra(SessionActivity.SESSION_ACCOUNT, teamId);
                    intent.putExtra(SessionActivity.SESSION_TYPE, SessionTypeEnum.Team);
                    startActivity(intent);
                    finish();
                }

                @Override
                public void onFailed(int code) {
                    UIUtils.showToast("" + code);
                }

                @Override
                public void onException(Throwable exception) {
                    UIUtils.showToast("");
                    exception.printStackTrace();
                }
            });
        }
    }

    @Override
    public void onFailed(int code) {
        UIUtils.showToast("" + code);
    }

    @Override
    public void onException(Throwable exception) {
        UIUtils.showToast("");
```

```
                exception.printStackTrace();
            }
        });
    }
}

@Override
public void onScanQRCodeOpenCameraError() {
    UIUtils.showToast("");
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == ImagePicker.RESULT_CODE_ITEMS) {//
        if (data != null) {
            //
//              boolean isOrig = data.getBooleanExtra(ImagePreviewActivity.ISORIGIN, false);
            final ArrayList<ImageItem> images = (ArrayList<ImageItem>)
data.getSerializableExtra(ImagePicker.EXTRA_RESULT_ITEMS);
            if (images != null && images.size() > 0) {
                //
                ThreadPoolFactory.getNormalPool().execute(new Runnable() {
                    @Override
                    public void run() {
                        String result = QRCodeDecoder.syncDecodeQRCode(images.get(0).path);
                        if (TextUtils.isEmpty(result)) {
                            UIUtils.showToast("");
                        } else {
                            handleResult(result);
                        }
                    }
                });
            }
        }
    }
}
```

35:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\SearchUserActivity.java

```
package com.lqr.wechat.activity;
```

```
import android.content.Intent;
import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.view.KeyEvent;
import android.view.MenuItem;
import android.view.View;
import android.view.inputmethod.EditorInfo;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.KeyBoardUtils;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class SearchUserActivity extends BaseActivity {

    private NimUserInfo mUser;

    public static final String SEARCH_TYPE = "search_type";
    public boolean isSearchUserLocal = SEARCH_USER_LOCAL;
    public static final boolean SEARCH_USER_LOCAL = true;
    public static final boolean SEARCH_USER_REMOTE = false;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
```

```
    @InjectView(R.id.etSearch)
    EditText mEtSearch;

    @InjectView(R.id.rlNoResultTip)
    RelativeLayout mRlNoResultTip;
    @InjectView(R.id.llSearch)
    LinearLayout mLlSearch;
    @InjectView(R.id.tvMsg)
    TextView mTvMsg;

    @Override
    public void init() {
        isSearchUserLocal = getIntent().getBooleanExtra(SEARCH_TYPE,
SEARCH_USER_LOCAL);
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_search_user);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public void initListener() {
        mEtSearch.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                mRlNoResultTip.setVisibility(View.GONE);
                if (TextUtils.isEmpty(mEtSearch.getText().toString().trim())) {
                    mLlSearch.setVisibility(View.GONE);
                } else {
                    mLlSearch.setVisibility(View.VISIBLE);
                    mTvMsg.setText(s);
                }
            }
```

```
            @Override
            public void afterTextChanged(Editable s) {

            }
        });

        //
        mEtSearch.setOnEditorActionListener(new TextView.OnEditorActionListener() {
            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
                if (actionId == EditorInfo.IME_ACTION_SEND || (event != null && event.getKeyCode()
== KeyEvent.KEYCODE_ENTER)) {
                    if (TextUtils.isEmpty(mEtSearch.getText().toString().trim())) {
                        KeyBoardUtils.closeKeybord(mEtSearch, SearchUserActivity.this);
                    } else {
                        doSearch();
                    }
                    return true;
                }
                return false;
            }
        });

        mLlSearch.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                doSearch();
            }
        });
    }

    private void doSearch() {
        showWaitingDialog("");
        String account = mEtSearch.getText().toString().trim();
        if (isSearchUserLocal) {
            mUser = NimUserInfoSDK.getUser(account);
            searchDone();
        } else {
            NimUserInfoSDK.getUserInfoFromServer(account, new
RequestCallback<List<NimUserInfo>>() {
                @Override
                public void onSuccess(List<NimUserInfo> param) {
                    if (param != null && param.size() > 0) {
```

```
                mUser = param.get(0);
                searchDone();
            }
        }

        @Override
        public void onFailed(int code) {
            UIUtils.showToast("" + code);
            hideWaitingDialog();
        }

        @Override
        public void onException(Throwable exception) {
            exception.printStackTrace();
            hideWaitingDialog();
        }
    });
  }
}

private void searchDone() {
  hideWaitingDialog();
  if (mUser == null) {
    mRlNoResultTip.setVisibility(View.VISIBLE);
  } else {
    mRlNoResultTip.setVisibility(View.GONE);
    //
    Intent intent = new Intent(this, UserInfoActivity.class);
    intent.putExtra("account", mUser.getAccount());
    startActivity(intent);
  }
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case android.R.id.home:
      finish();
      break;
  }
  return super.onOptionsItemSelected(item);
}
```

```
    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
        mEtSearch.setVisibility(View.VISIBLE);
        mEtSearch.setHintTextColor(UIUtils.getColor(R.color.gray2));
        mEtSearch.setTextColor(UIUtils.getColor(R.color.white));
    }

}
```

36:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\SessionActivity.java

```
package com.lqr.wechat.activity;

import android.Manifest;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.os.SystemClock;
import android.support.v4.view.ViewPager;
import android.support.v7.widget.GridLayoutManager;
import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.Chronometer;
import android.widget.EditText;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
```

```
import com.lqr.emoji.EmoticonPickerView;
import com.lqr.emoji.EmotionKeyboard;
import com.lqr.emoji.IEmoticonSelectedListener;
import com.lqr.imagepicker.ImagePicker;
import com.lqr.imagepicker.bean.ImageItem;
import com.lqr.imagepicker.ui.ImagePreviewActivity;
import com.lqr.recyclerview.LQRRecyclerView;
import com.lqr.videorecordview.LQRVideoRecordView;
import com.lqr.wechat.R;
import com.lqr.wechat.adapter.FuncPagerAdapter;
import com.lqr.wechat.adapter.SessionAdapter;
import com.lqr.wechat.factory.ThreadPoolFactory;
import com.lqr.wechat.fragment.BaseFragment;
import com.lqr.wechat.fragment.Func1Fragment;
import com.lqr.wechat.fragment.Func2Fragment;
import com.lqr.wechat.model.Contact;
import com.lqr.wechat.nimsdk.NimHistorySDK;
import com.lqr.wechat.nimsdk.NimMessageSDK;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.nimsdk.custom.StickerAttachment;
import com.lqr.wechat.nimsdk.helper.SendImageHelper;
import com.lqr.wechat.utils.KeyBoardUtils;
import com.lqr.wechat.utils.LogUtils;
import com.lqr.wechat.utils.UIUtils;
import com.lqr.wechat.view.DotView;
import com.lqr.wechat.view.LQRRecordProgress;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.ResponseCode;
import com.netease.nimlib.sdk.media.record.AudioRecorder;
import com.netease.nimlib.sdk.media.record.IAudioRecordCallback;
import com.netease.nimlib.sdk.media.record.RecordType;
import com.netease.nimlib.sdk.msg.MessageBuilder;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.msg.model.AttachmentProgress;
import com.netease.nimlib.sdk.msg.model.IMMessage;
import com.netease.nimlib.sdk.msg.model.QueryDirectionEnum;
import com.netease.nimlib.sdk.team.model.Team;
import com.netease.nimlib.sdk.team.model.TeamMember;
```

```
import java.io.File;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;
import butterknife.OnTouch;
import cn.bingoogolapple.refreshlayout.BGANormalRefreshViewHolder;
import cn.bingoogolapple.refreshlayout.BGARefreshLayout;
import cn.bingoogolapple.refreshlayout.BGARefreshViewHolder;
import kr.co.namee.permissiongen.PermissionFail;
import kr.co.namee.permissiongen.PermissionGen;
import kr.co.namee.permissiongen.PermissionSuccess;

/**
 * @ CSDN_LQR
 * @
 * <p>
 *
 */
public class SessionActivity extends BaseActivity implements IEmoticonSelectedListener,
BGARefreshLayout.BGARefreshLayoutDelegate, IAudioRecordCallback,
LQRVideoRecordView.OnRecordStausChangeListener {

    public static final int IMAGE_PICKER = 100;

    public static final String SESSION_ACCOUNT = "account";
    public static final String SESSION_TYPE = "type";

    //
    public String mSessionId;//idid
    private Contact mContact;
    private Team mTeam;

    public SessionTypeEnum mSessionType = SessionTypeEnum.P2P;
    //
    private Observer<IMMessage> mMsgStatusObserver;
    private Observer<List<IMMessage>> mIncomingMessageObserver;
    private Observer<AttachmentProgress> mAttachmentProgressObserver;
    private List<IMMessage> mMessages = new ArrayList<>();
```

```
private SessionAdapter mAdapter;

private Runnable mCvMessageScrollToBottomTask = new Runnable() {
    @Override
    public void run() {
        mCvMessage.moveToPosition(mMessages.size() - 1);
    }
};
//
private IMMessage mAnchor;
private QueryDirectionEnum mDirection = QueryDirectionEnum.QUERY_OLD;//
private static final int LOAD_MESSAGE_COUNT = 20;
private boolean mFirstLoad = true;

private boolean mRemote = false;
//
private FuncPagerAdapter mBottomFucAdapter;
private List<BaseFragment> mFragments;

private EmotionKeyboard mEmotionKeyboard;
//
private AudioRecorder mAudioRecorderHelper;
private boolean mStartRecord;
private boolean mCanclled;

private boolean mTouched;

@InjectView(R.id.toolbar)
Toolbar mToolbar;
@InjectView(R.id.refreshLayout)
BGARefreshLayout mRefreshLayout;

@InjectView(R.id.cvMessage)
LQRRecyclerView mCvMessage;

@InjectView(R.id.llButtomFunc)
LinearLayout mLlButtomFunc;
@InjectView(R.id.ivAudio)
ImageView mIvAudio;
@InjectView(R.id.etContent)
EditText mEtContent;
@InjectView(R.id.btnAudio)
```

```
Button mBtnAudio;
@InjectView(R.id.ivEmo)
ImageView mIvEmo;
@InjectView(R.id.ivAdd)
ImageView mIvAdd;

@InjectView(R.id.btnSend)
Button mBtnSend;
@InjectView(R.id.flBottom)
FrameLayout mFlBottom;
@InjectView(R.id.epv)
EmoticonPickerView mEpv;
@InjectView(R.id.vpFunc)
ViewPager mVpFunc;

@InjectView(R.id.dv)
DotView mDv;
@InjectView(R.id.flPlayAudio)
FrameLayout mFlPlayAudio;
@InjectView(R.id.cTimer)
Chronometer mCTimer;

@InjectView(R.id.tvTimerTip)
TextView mTvTimerTip;
@InjectView(R.id.llPlayVideo)
LinearLayout mLlPlayVideo;
@InjectView(R.id.vrvVideo)
LQRVideoRecordView mVrvVideo;
@InjectView(R.id.tvTipOne)
TextView mTvTipOne;
@InjectView(R.id.tvTipTwo)
TextView mTvTipTwo;
@InjectView(R.id.rp)
LQRRecordProgress mRp;
@InjectView(R.id.btnVideo)
Button mBtnVideo;
private Observer<TeamMember> memberRemoveObserver;
private Observer<List<TeamMember>> memberUpdateObserver;

@OnTouch(R.id.cvMessage)
public boolean cvTouch() {
    if (mEtContent.hasFocus()) {
```

```
            closeKeyBoardAndLoseFocus();
            return true;
        } else if (mFlBottom.getVisibility() == View.VISIBLE) {
            mFlBottom.setVisibility(View.GONE);
            closeKeyBoardAndLoseFocus();
            return true;
        }
        return false;
    }

    @OnClick({R.id.ivAudio, R.id.btnSend})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.ivAudio:
                toggleAudioButtonVisibility();
                break;
            case R.id.btnSend:
                sendTextMsg();
                break;
        }
    }

    @Override
    public void init() {
        Intent intent = getIntent();
        SessionTypeEnum sessionType = (SessionTypeEnum)
intent.getSerializableExtra(SESSION_TYPE);
        if (sessionType != null) {
            mSessionType = sessionType;
        }

        mSessionId = intent.getStringExtra(SESSION_ACCOUNT);
        if (TextUtils.isEmpty(mSessionId)) {
            interrupt();
            return;
        }

        registerAllObserver();
        requestPermission();
    }

    @Override
```

```
    public void initView() {
        setContentView(R.layout.activity_session);
        ButterKnife.inject(this);
        initToolbar();
        initEmotionPickerView();
        initEmotionKeyboard();
        initRefreshLayout();
        initBottomFunc();

        //RecyclerView
//        ((DefaultItemAnimator)
mCvMessage.getItemAnimator()).setSupportsChangeAnimations(false);

        closeKeyBoardAndLoseFocus();
    }

    @Override
    public void initData() {
        //()
        mMessages.clear();
        setAdapter();
        loadHistoryMsgFromLocal();

        if (mSessionType == SessionTypeEnum.P2P) {
            mContact = new Contact(mSessionId);
            ///
            getSupportActionBar().setTitle(TextUtils.isEmpty(mContact.getAlias()) ?
mContact.getName() : mContact.getAlias());
        } else {
            //
            ThreadPoolFactory.getNormalPool().execute(new Runnable() {
                @Override
                public void run() {
                    mTeam = NimTeamSDK.queryTeamBlock(mSessionId);
                    UIUtils.postTaskSafely(new Runnable() {
                        @Override
                        public void run() {
                            getSupportActionBar().setTitle(TextUtils.isEmpty(mTeam.getName()) ? "(" +
mTeam.getMemberCount() + ")" : mTeam.getName());
                        }
                    });
                }
```

```
        });

    }
}


@Override
public void initListener() {
    //
    mEtContent.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {

        }

        @Override
        public void afterTextChanged(Editable s) {
            if (TextUtils.isEmpty(mEtContent.getText().toString())) {
                mIvAdd.setVisibility(View.VISIBLE);
                mBtnSend.setVisibility(View.GONE);
            } else {
                mIvAdd.setVisibility(View.GONE);
                mBtnSend.setVisibility(View.VISIBLE);
            }
        }
    });

    //
    mEtContent.setOnFocusChangeListener(new View.OnFocusChangeListener() {
        @Override
        public void onFocusChange(View v, boolean hasFocus) {
            if (hasFocus) {
                cvScrollToBottom();
            }
        }
    });
```

```
        //ViewPager
        mVpFunc.setOnPageChangeListener(new ViewPager.OnPageChangeListener() {
            @Override
            public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {

            }

            @Override
            public void onPageSelected(int position) {
                //
                mDv.changeCurrentPage(position);
            }

            @Override
            public void onPageScrollStateChanged(int state) {

            }
        });

        //
//        " "
//        " "
//        ""
        mBtnAudio.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                switch (event.getAction()) {
                    case MotionEvent.ACTION_DOWN:
                        mTouched = true;
                        initAudioRecord();
                        onStartAudioRecord();
                        break;
                    case MotionEvent.ACTION_MOVE:
                        mTouched = false;
                        cancelAudioRecord(isCancelled(v, event));
                        break;
                    case MotionEvent.ACTION_UP:
                        mTouched = false;
                        hidePlayAudio();
                        onEndAudioRecord(isCancelled(v, event));
                        break;
                    case MotionEvent.ACTION_CANCEL:
```

```
                mTouched = false;
                hidePlayAudio();
                onEndAudioRecord(isCancelled(v, event));
                break;
            }
            return false;
        }
    });

    //
    mBtnVideo.setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    mRp.start();
                    mRp.setProgressColor(Color.parseColor("#1AAD19"));
                    mTvTipOne.setVisibility(View.VISIBLE);
                    mTvTipTwo.setVisibility(View.GONE);
                    //
                    mVrvVideo.record(SessionActivity.this);
                    break;
                case MotionEvent.ACTION_UP:
                    mRp.stop();
                    mTvTipOne.setVisibility(View.GONE);
                    mTvTipTwo.setVisibility(View.GONE);
                    //
                    if (mVrvVideo.getTimeCount() > 3) {
                        if (!isCancelled(v, event)) {
                            onRecrodFinish();
                        } else {
                            if (mVrvVideo.getVecordFile() != null)
                                mVrvVideo.getVecordFile().delete();
                        }
                    } else {
                        if (!isCancelled(v, event)) {
                            Toast.makeText(getApplicationContext(), "", Toast.LENGTH_SHORT).show();
                        } else {
                            if (mVrvVideo.getVecordFile() != null)
                                mVrvVideo.getVecordFile().delete();
                        }
                    }
```

```
                resetVideoRecord();
                break;
            case MotionEvent.ACTION_MOVE:
                if (isCancelled(v, event)) {
                    mTvTipOne.setVisibility(View.GONE);
                    mTvTipTwo.setVisibility(View.VISIBLE);
                    mRp.setProgressColor(Color.parseColor("#FF1493"));
                } else {
                    mTvTipOne.setVisibility(View.VISIBLE);
                    mTvTipTwo.setVisibility(View.GONE);
                    mRp.setProgressColor(Color.parseColor("#1AAD19"));
                }
                break;
        }
        return true;
      }
    });
  }

  @Override
  public void onResume() {
    if (mSessionType == SessionTypeEnum.Team) {
      mTeam = NimTeamSDK.queryTeamBlock(mSessionId);
      getSupportActionBar().setTitle(TextUtils.isEmpty(mTeam.getName()) ? "(" +
mTeam.getMemberCount() + ")" : mTeam.getName());
    }
    setAdapter();
    super.onResume();
  }

  @Override
  public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
  }


  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    new MenuInflater(this).inflate(R.menu.menu_info, menu);
    return true;
  }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                onBackPressed();
                break;
            case R.id.itemFriendInfo:
                Intent intent;
                if (mSessionType == SessionTypeEnum.P2P) {
                    intent = new Intent(SessionActivity.this, UserInfoActivity.class);
                    intent.putExtra(UserInfoActivity.USER_INFO_ACCOUNT, mSessionId);
                    startActivity(intent);
                } else {
                    intent = new Intent(SessionActivity.this, TeamCheatInfoActivity.class);
                    intent.putExtra(TeamCheatInfoActivity.GROUP_CHEAT_INFO_TEAMID,
mSessionId);
                    startActivityForResult(intent, 100);
                }
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == ImagePicker.RESULT_CODE_ITEMS) {//
            if (data != null) {
                //
                boolean isOrig = data.getBooleanExtra(ImagePreviewActivity.ISORIGIN, false);
                ArrayList<ImageItem> images = (ArrayList<ImageItem>)
data.getSerializableExtra(ImagePicker.EXTRA_RESULT_ITEMS);

                for (ImageItem imageItem : images) {
                    new SendImageHelper.SendImageTask(SessionActivity.this, isOrig, imageItem, new
SendImageHelper.Callback() {
                        @Override
                        public void sendImage(File file, boolean isOrig) {
                            sendImagesMsg(file);
                        }
                    }).execute();
                }
```

```
            }
        } else if (resultCode == TeamCheatInfoActivity.RESP_QUIT_TEAM || resultCode ==
TeamCheatInfoActivity.RESP_CHEAT_SINGLE) {
            finish();
        } else if (resultCode ==
TeamCheatInfoActivity.RESP_CLEAR_CHATTING_RECORD_HISTORY) {
            mAdapter.clearData();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        //
        unRegisterAllObserver();
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }

    private void initRefreshLayout() {
        // BGARefreshLayout
        mRefreshLayout.setDelegate(this);
        //      12
        BGARefreshViewHolder refreshViewHolder = new BGANormalRefreshViewHolder(this,
false);
        //
        mRefreshLayout.setRefreshViewHolder(refreshViewHolder);
    }

    /**
     *
     */
    private void initBottomFunc() {
        //
        mFragments = new ArrayList<>();
        Func1Fragment func1Fragment1 = new Func1Fragment();
        Func2Fragment func1Fragment2 = new Func2Fragment();
```

```
        mFragments.add(func1Fragment1);
        mFragments.add(func1Fragment2);
        mBottomFucAdapter = new FuncPagerAdapter(getSupportFragmentManager(),
mFragments);
        mVpFunc.setAdapter(mBottomFucAdapter);

        //
        mDv.initData(mFragments.size(), 0);
    }

    public void setAdapter() {
        if (mAdapter == null) {
            mAdapter = new SessionAdapter(this, mMessages);
            mCvMessage.setAdapter(mAdapter);
        } else {
            mAdapter.notifyDataSetChanged();
        }
    }

    private void registerAllObserver() {
        observeMsgStatus();
        observeReceiveMessage();
        observerAttachmentProgressObserver();
        if (mSessionType == SessionTypeEnum.Team) {
            observeMemberRemove();
            observeMemberUpdate();
        }
    }

    private void unRegisterAllObserver() {
        NimMessageSDK.observeMsgStatus(mMsgStatusObserver, false);
        NimMessageSDK.observeReceiveMessage(mIncomingMessageObserver, false);
        NimMessageSDK.observeAttachProgress(mAttachmentProgressObserver, false);
        if (mSessionType == SessionTypeEnum.Team) {
            NimTeamSDK.observeMemberRemove(memberRemoveObserver, false);
            NimTeamSDK.observeMemberUpdate(memberUpdateObserver, false);
        }
    }

    /**
     *
     */
```

```
    private void observeMsgStatus() {
        mMsgStatusObserver = new Observer<IMMessage>() {
            @Override
            public void onEvent(IMMessage imMessage) {
                if (NimMessageSDK.isCurrentSessionMessage(imMessage, mSessionId,
mSessionType)) {
                    onMessageStatusChange(imMessage);
                }
            }
        };
        NimMessageSDK.observeMsgStatus(mMsgStatusObserver, true);
    }

    /**
     *
     */
    private void observeReceiveMessage() {
        mIncomingMessageObserver = new Observer<List<IMMessage>>() {
            @Override
            public void onEvent(List<IMMessage> imMessages) {
                if (imMessages == null || imMessages.isEmpty()) {
                    return;
                }

                //
                List<IMMessage> currentMsgList = new ArrayList<>();
                for (IMMessage msg : imMessages) {
                    if (NimMessageSDK.isCurrentSessionMessage(msg, mSessionId, mSessionType)) {
                        currentMsgList.add(msg);
                    }
                }

                //
                int theLastOnePosition = mAdapter.getData().size() - 1;
                mAdapter.addMoreData(currentMsgList);

                //
                int lastVisibleItemPosition = ((GridLayoutManager)
mCvMessage.getLayoutManager()).findLastVisibleItemPosition();
                if (lastVisibleItemPosition == theLastOnePosition)
                    cvScrollToBottom();
```

```
        }
    };
    NimMessageSDK.observeReceiveMessage(mIncomingMessageObserver, true);
}

/**
 * /
 */
private void observerAttachmentProgressObserver() {
    mAttachmentProgressObserver = new Observer<AttachmentProgress>() {
        @Override
        public void onEvent(AttachmentProgress progress) {
            onAttachmentProgressChange(progress);
        }
    };
    NimMessageSDK.observeAttachProgress(mAttachmentProgressObserver, true);
}

private void observeMemberUpdate() {
    memberUpdateObserver = new Observer<List<TeamMember>>() {
        @Override
        public void onEvent(List<TeamMember> teamMembers) {
            onResume();
        }
    };
    NimTeamSDK.observeMemberUpdate(memberUpdateObserver, true);
}

private void observeMemberRemove() {
    memberRemoveObserver = new Observer<TeamMember>() {
        @Override
        public void onEvent(TeamMember teamMember) {
            onResume();
        }
    };
    NimTeamSDK.observeMemberRemove(memberRemoveObserver, true);
}

private void onMessageStatusChange(IMMessage message) {
    int index = getItemIndex(message.getUuid());
    if (index >= 0 && index < mMessages.size()) {
        IMMessage msg = mMessages.get(index);
```

```
            msg.setStatus(message.getStatus());
            msg.setAttachStatus(message.getAttachStatus());
            mAdapter.notifyItemChanged(index);
        }
    }

    private void onAttachmentProgressChange(AttachmentProgress progress) {
        int index = getItemIndex(progress.getUuid());
        if (index >= 0 && index < mMessages.size()) {
            IMMessage item = mMessages.get(index);
            LogUtils.sf("Transferred = " + progress.getTransferred());
            LogUtils.sf("Total = " + progress.getTotal());
            float value = (float) progress.getTransferred() / (float) progress.getTotal();
            mAdapter.putProgress(item, value * 100);
            mAdapter.notifyItemChanged(index);
        }
    }

    private int getItemIndex(String uuid) {
        for (int i = 0; i < mMessages.size(); i++) {
            IMMessage message = mMessages.get(i);
            if (TextUtils.equals(message.getUuid(), uuid)) {
                return i;
            }
        }
        return -1;
    }



    /**
     *
     */
    private IMMessage getAnchor() {
        if (mMessages.size() == 0) {
            return mAnchor == null ? MessageBuilder.createEmptyMessage(mSessionId,
mSessionType, 0) : mAnchor;
        } else {
            int index = (mDirection == QueryDirectionEnum.QUERY_NEW ? mMessages.size() - 1 :
0);
            return mMessages.get(index);
        }
    }
```

```
    /**
     *
     */
    private void loadHistoryMsgFromLocal() {
        LogUtils.sf("");
        mDirection = QueryDirectionEnum.QUERY_OLD;
        mRemote = false;
        NimHistorySDK.queryMessageListEx(getAnchor(), mDirection, LOAD_MESSAGE_COUNT,
true).setCallback(loadFromRemoteCallback);
    }

    /**
     *
     */
//    private void loadNewMsgFromServer() {
//        LogUtils.sf("");
//        mDirection = QueryDirectionEnum.QUERY_NEW;
//        mRemote = true;
//        NimHistorySDK.pullMessageHistoryEx(getAnchor(), new
DateTime(2017,1,5,23,59,59).getMillis(), LOAD_MESSAGE_COUNT, mDirection,
true).setCallback(loadFromRemoteCallback);
//    }

    /**
     *
     */
    private void loadHistoryMsgFromRemote() {
        LogUtils.sf("");
        mDirection = QueryDirectionEnum.QUERY_OLD;
        mRemote = true;
        NimHistorySDK.pullMessageHistory(getAnchor(), LOAD_MESSAGE_COUNT,
true).setCallback(loadFromRemoteCallback);
    }

    private boolean mIsFirstLoadHistory = true;

    RequestCallback<List<IMMessage>> loadFromRemoteCallback = new
RequestCallbackWrapper<List<IMMessage>>() {
        @Override
        public void onResult(int code, List<IMMessage> result, Throwable exception) {
            if (code != ResponseCode.RES_SUCCESS || exception != null) {
```

```
            return;
        }

        if (result == null)
            return;

        //
        if (mIsFirstLoadHistory) {
            mIsFirstLoadHistory = false;
        }
        //
        else if (result.size() == 0 && !mRemote) {
            loadHistoryMsgFromRemote();
            return;
        }

        onMessageLoaded(result);
    }
};



/**
 *
 *
 * @param messages
 */
private void onMessageLoaded(List<IMMessage> messages) {
    if (mRemote) {
        Collections.reverse(messages);
    }

    if (mFirstLoad && mMessages.size() > 0) {
        //
        for (IMMessage message : messages) {
            for (IMMessage item : mMessages) {
                if (item.isTheSame(message)) {
                    mAdapter.removeItem(item);
                    break;
                }
            }
        }
    }
```

```
        if (mFirstLoad && mAnchor != null) {
            mAdapter.addLastItem(mAnchor);
        }

        if (mDirection == QueryDirectionEnum.QUERY_NEW) {
            mAdapter.addMoreData(messages);
        } else {
            mAdapter.addNewData(messages);
        }

        if (mFirstLoad) {
            cvScrollToBottom();
        } else {
            if (messages.size() > 0) {
                mCvMessage.moveToPosition(messages.size() - 1);
            }
        }

        mRefreshLayout.endRefreshing();

        mFirstLoad = false;
    }

    /**
     *
     */
    public void sendTextMsg() {
        String content = mEtContent.getText().toString();
        if (!TextUtils.isEmpty(content)) {
            IMMessage message = NimMessageSDK.createTextMessage(mSessionId,
mSessionType, content);
            sendMsg(message);
            mEtContent.setText("");
        }
    }

    /**
     *
     *
     * @param stickerAttachment
     */
```

```
    private void sendStickerMsg(StickerAttachment stickerAttachment) {
        IMMessage stickerMessage = NimMessageSDK.createCustomMessage(mSessionId,
mSessionType, "", stickerAttachment);
        sendMsg(stickerMessage);
    }

    /**
     *
     */
    private void sendImagesMsg(File image) {
        IMMessage message = NimMessageSDK.createImageMessage(mSessionId,
mSessionType, image.getAbsoluteFile(), image.getName());
        sendMsg(message);
    }

    /**
     *
     */
    private void sendAudioMsg(File audioFile, long audioLength) {
        IMMessage msg = NimMessageSDK.createAudioMessage(mSessionId, mSessionType,
audioFile, audioLength);
        sendMsg(msg);
    }

    /**
     *
     */
    private void sendVidoMsg(File videoFile, String displayName) {
        IMMessage msg = NimMessageSDK.createVideoMessage(mSessionId, mSessionType,
videoFile, displayName);
        sendMsg(msg);
    }


    /**
     *
     */
    private void sendMsg(IMMessage message) {
        NimMessageSDK.sendMessage(message);
        mAdapter.addLastItem(message);
        mAdapter.notifyDataSetChanged();
        cvScrollToBottom();
```

```
}

/**
 *
 */
private void openKeyBoardAndGetFocus() {
    mEtContent.requestFocus();
    KeyBoardUtils.openKeybord(mEtContent, this);
}

/**
 *
 */
private void closeKeyBoardAndLoseFocus() {
    mEtContent.clearFocus();
    KeyBoardUtils.closeKeybord(mEtContent, this);
    mFlBottom.setVisibility(View.GONE);
}

/**
 *
 */
private void cvScrollToBottom() {
    UIUtils.postTaskDelay(mCvMessageScrollToBottomTask, 100);
}

/*==================  begin ==================*/

/**
 *
 */
private void initEmotionPickerView() {
    mEpv.setWithSticker(true);
    mEpv.show(this);
    mEpv.attachEditText(mEtContent);
}

/**
 *
 */
private void initEmotionKeyboard() {
    //1EmotionKeyboard
```

```
        mEmotionKeyboard = EmotionKeyboard.with(this);
        //2
        mEmotionKeyboard.bindToEditText(mEtContent);
        //3RecyclerView
        mEmotionKeyboard.bindToContent(mCvMessage);
        //4FrameLayoutFrameLayout
        mEmotionKeyboard.setEmotionView(mFlBottom);
        //52
        mEmotionKeyboard.bindToEmotionButton(mIvEmo, mIvAdd);
        //65EmotionButtonviewEmotionKeyboard
        mEmotionKeyboard.setOnEmotionButtonOnClickListener(new
EmotionKeyboard.OnEmotionButtonOnClickListener() {
            @Override
            public boolean onEmotionButtonOnClickListener(View view) {
                if (mBtnAudio.getVisibility() == View.VISIBLE) {
                    hideBtnAudio();
                }
                //
                if (mFlBottom.getVisibility() == View.VISIBLE) {
                    //ivAdd
                    if (mEpv.getVisibility() == View.VISIBLE && view.getId() == R.id.ivAdd) {
                        mEpv.setVisibility(View.GONE);
                        mLlButtomFunc.setVisibility(View.VISIBLE);
                        return true;
                        //ivEmo
                    } else if (mLlButtomFunc.getVisibility() == View.VISIBLE && view.getId() ==
R.id.ivEmo) {
                        mEpv.setVisibility(View.VISIBLE);
                        mLlButtomFunc.setVisibility(View.GONE);
                        return true;
                    }
                } else {
                    //ivEmo
                    if (view.getId() == R.id.ivEmo) {
                        mEpv.setVisibility(View.VISIBLE);
                        mLlButtomFunc.setVisibility(View.GONE);
                        //ivAdd
                    } else {
                        mEpv.setVisibility(View.GONE);
                        mLlButtomFunc.setVisibility(View.VISIBLE);
                    }
                }
```

```
                cvScrollToBottom();
                return false;
            }
        });
    }

    @Override
    public void onEmojiSelected(String s) {
    }

    @Override
    public void onStickerSelected(String catalog, String chartlet) {
        StickerAttachment stickerAttachment = new StickerAttachment(catalog, chartlet);
        sendStickerMsg(stickerAttachment);
    }

    /*==================  end ==================*/
    /*==================  begin ==================*/

    /**
     *
     */
    public void toggleAudioButtonVisibility() {
        if (mBtnAudio.getVisibility() == View.VISIBLE) {
            hideBtnAudio();
        } else {
            showBtnAudio();
        }
        //
        mIvAudio.setImageResource(mBtnAudio.getVisibility() == View.VISIBLE ?
R.mipmap.ic_cheat_keyboard : R.mipmap.ic_cheat_voice);
    }

    private void showBtnAudio() {
        mBtnAudio.setVisibility(View.VISIBLE);
        mEtContent.setVisibility(View.GONE);
        mIvEmo.setVisibility(View.GONE);
        //
        closeKeyBoardAndLoseFocus();
    }

    private void hideBtnAudio() {
```

```
    mBtnAudio.setVisibility(View.GONE);
    mEtContent.setVisibility(View.VISIBLE);
    mIvEmo.setVisibility(View.VISIBLE);
    //
    openKeyBoardAndGetFocus();
}

private void showPlayAudio() {
    mBtnAudio.setText(" ");
    mBtnAudio.setBackgroundResource(R.drawable.shape_btn_voice_press);
}

private void hidePlayAudio() {
    mBtnAudio.setText(" ");
    mBtnAudio.setBackgroundResource(R.drawable.shape_btn_voice_normal);
    mFlPlayAudio.setVisibility(View.GONE);
}

/**
 *
 */
private void updateTimerTip(boolean cancel) {
    if (cancel) {
        mTvTimerTip.setText("");
        mTvTimerTip.setBackgroundResource(R.drawable.shape_bottom_corner_solid_red);
        mBtnAudio.setText("");
    } else {
        mTvTimerTip.setText("");
        mTvTimerTip.setBackgroundResource(0);
        mBtnAudio.setText(" ");
    }
}

/**
 *
 */
private void startAudioRecordAnim() {
    mFlPlayAudio.setVisibility(View.VISIBLE);
    mCTimer.setBase(SystemClock.elapsedRealtime());//
    mCTimer.start();
}
```

```
    /**
     *
     */
    private void stopAudiioRecordAnim() {
        mFlPlayAudio.setVisibility(View.GONE);
        mCTimer.stop();
        mCTimer.setBase(SystemClock.elapsedRealtime());//
    }

    private static boolean isCancelled(View view, MotionEvent event) {
        int[] location = new int[2];
        view.getLocationOnScreen(location);

        if (event.getRawX() < location[0] || event.getRawX() > location[0] + view.getWidth()
                || event.getRawY() < location[1] - 40) {
            return true;
        }

        return false;
    }

    /**
     * AudioRecord
     */
    private void initAudioRecord() {
        if (mAudioRecorderHelper == null)
            mAudioRecorderHelper = new AudioRecorder(this, RecordType.AAC,
AudioRecorder.DEFAULT_MAX_AUDIO_RECORD_TIME_SECOND, this);
    }

    /**
     *
     */
    private void onStartAudioRecord() {
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON,
WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
        mStartRecord = mAudioRecorderHelper.startRecord();
        mCanclled = false;
        if (mStartRecord == false) {
            UIUtils.showToast("");
            return;
        }
```

```
    if (!mTouched) {
        return;
    }

    showPlayAudio();
    updateTimerTip(false);
    startAudioRecordAnim();
}

/**
 *
 */
private void onEndAudioRecord(boolean cancel) {
    getWindow().setFlags(0, WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    mAudioRecorderHelper.completeRecord(cancel);
    hidePlayAudio();
    stopAudiioRecordAnim();
}

/**
 *
 */
private void cancelAudioRecord(boolean cancel) {
    if (!mStartRecord) {
        return;
    }

    if (mCanclled == cancel) {
        return;
    }

    mCanclled = cancel;
    updateTimerTip(cancel);
}

@Override
public void onRecordReady() {

}
```

```
@Override
public void onRecordStart(File audioFile, RecordType recordType) {

}

@Override
public void onRecordSuccess(File audioFile, long audioLength, RecordType recordType) {
    sendAudioMsg(audioFile, audioLength);
}

@Override
public void onRecordFail() {

}

@Override
public void onRecordCancel() {

}

@Override
public void onRecordReachedMaxTime(final int maxTime) {
    stopAudiioRecordAnim();
    showMaterialDialog("", "", "", "", new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mAudioRecorderHelper.handleEndRecord(true, maxTime);
            hideMaterialDialog();
        }
    }, new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            hideMaterialDialog();
        }
    });
}

public boolean isRecording() {
    return mAudioRecorderHelper != null && mAudioRecorderHelper.isRecording();
}

public void requestPermission() {
```

```
        PermissionGen.with(this)
                .addRequestCode(100)
                .permissions(
                        Manifest.permission.CAMERA,
                        Manifest.permission.RECORD_AUDIO)
                .request();
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions,
                                           int[] grantResults) {
        PermissionGen.onRequestPermissionsResult(this, requestCode, permissions, grantResults);
    }

    @PermissionSuccess(requestCode = 100)
    public void doSomething() {
//        UIUtils.showToast("");
    }

    @PermissionFail(requestCode = 100)
    public void doFailSomething() {
        UIUtils.showToast("");
    }

    /*==================  end ==================*/
    /*==================  begin ==================*/

    @Override
    public void onBackPressed() {
        if (mIsPlayVideoShown) {
            hidePlayVideo();
            return;
        }
        super.onBackPressed();
    }

    private boolean mIsPlayVideoShown = false;//

    public void showPlayVideo() {
        mLlPlayVideo.setVisibility(View.VISIBLE);
        initVideoRecord();
        mIsPlayVideoShown = true;
```

```
}

public void hidePlayVideo() {
    mLlPlayVideo.setVisibility(View.GONE);
    releaseVideoRecord();
    mIsPlayVideoShown = false;
    cvTouch();
}

public void initVideoRecord() {
    UIUtils.postTaskDelay(new Runnable() {
        @Override
        public void run() {
            mVrvVideo.openCamera();
        }
    }, 1000);
}

public void releaseVideoRecord() {
    mVrvVideo.stop();
}

/**
 *
 */
public void resetVideoRecord() {
    mVrvVideo.stop();
    mVrvVideo.openCamera();
}

@Override
public void onRecrodFinish() {
    UIUtils.postTaskSafely(new Runnable() {
        @Override
        public void run() {
            mTvTipOne.setVisibility(View.GONE);
            mTvTipTwo.setVisibility(View.GONE);
            resetVideoRecord();
            //
            sendVidoMsg(mVrvVideo.getVecordFile(), mVrvVideo.getVecordFile().getName());
        }
    });
```

```
    }

    @Override
    public void onRecording(int timeCount, int recordMaxTime) {

    }

    @Override
    public void onRecordStart() {

    }

    /*==================  end ==================*/
    /*==================  begin ==================*/
    @Override
    public void onBGARefreshLayoutBeginRefreshing(BGARefreshLayout refreshLayout) {
        loadHistoryMsgFromRemote();
    }

    @Override
    public boolean onBGARefreshLayoutBeginLoadingMore(BGARefreshLayout refreshLayout) {
        return false;
    }


    /*==================  end ==================*/
}
```

37:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\SettingActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;

import com.lqr.wechat.App;
import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimAccountSDK;
import com.lqr.wechat.view.CustomDialog;
```

```
import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class SettingActivity extends BaseActivity {

    Intent intent;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    private View mExitDialogView;
    private CustomDialog mDialog;

    @OnClick({R.id.oivNewMsgNotifySet, R.id.oivDontDistorbSet, R.id.oivCheatSet,
R.id.oivPrivacySet, R.id.oivCommon, R.id.oivAccountAndSafeSet, R.id.oivAbout, R.id.oivExit})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.oivNewMsgNotifySet:
                intent = new Intent(this, NewMsgNotifySetActivity.class);
                startActivity(intent);
                break;
            case R.id.oivDontDistorbSet:
                intent = new Intent(this, DontDistorbSetActivity.class);
                startActivity(intent);
                break;
            case R.id.oivCheatSet:
                intent = new Intent(this, CheatSetActivity.class);
                startActivity(intent);
                break;
            case R.id.oivPrivacySet:
                intent = new Intent(this, PrivacySetActivity.class);
                startActivity(intent);
                break;
            case R.id.oivCommon:
                intent = new Intent(this, CommonSetActivity.class);
                startActivity(intent);
                break;
            case R.id.oivAccountAndSafeSet:
```

```
            intent = new Intent(this, AccountAndSafeSetActivity.class);
            startActivity(intent);
            break;
        case R.id.oivAbout:
            intent = new Intent(this, AboutActivity.class);
            startActivity(intent);
            break;
        case R.id.oivExit:
            if (mExitDialogView == null) {
                mExitDialogView = View.inflate(this, R.layout.dialog_exit, null);
                mDialog = new CustomDialog(this, mExitDialogView, R.style.dialog);
                mDialog.show();

                mExitDialogView.findViewById(R.id.tvExitAccount).setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        //
                        NimAccountSDK.logout();
                        intent = new Intent(SettingActivity.this, LoginActivity.class);
                        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
                        startActivity(intent);
                        finish();
                        mDialog.dismiss();
                    }
                });

                mExitDialogView.findViewById(R.id.tvExitApp).setOnClickListener(new
View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        //app
                        App.exit();
                        mDialog.dismiss();
                    }
                });

            } else {
                mDialog.show();
            }
            break;
```

```
        }
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_setting);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

38:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\ShowBigImageActivity.java

```
package com.lqr.wechat.activity;

import android.os.Environment;
import android.support.v7.widget.Toolbar;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.FrameLayout;
import android.widget.PopupWindow;
```

```
import android.widget.ProgressBar;
import android.widget.TextView;

import com.bm.library.PhotoView;
import com.lqr.wechat.R;
import com.lqr.wechat.factory.PopupWindowFactory;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.utils.UIUtils;
import com.zhy.http.okhttp.OkHttpUtils;
import com.zhy.http.okhttp.callback.FileCallBack;

import java.io.File;

import butterknife.ButterKnife;
import butterknife.InjectView;
import okhttp3.Call;

/**
 * @ CSDN_LQR
 * @
 */
public class ShowBigImageActivity extends BaseActivity {

    private String mUrl;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.pv)
    PhotoView mPv;
    @InjectView(R.id.pb)
    ProgressBar mPb;
    private FrameLayout mView;
    private PopupWindow mPopupWindow;

    @Override
    public void init() {
        mUrl = getIntent().getStringExtra("url");
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_show_big_image);
```

```
        ButterKnife.inject(this);
        initToolbar();
        mPv.enable();//

        ImageLoaderManager.LoadNetImage(mUrl, mPv);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        new MenuInflater(this).inflate(R.menu.menu_more, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
            case R.id.itemMore:
                showPopupMenu();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }

    private void showPopupMenu() {
        if (mView == null) {
            mView = new FrameLayout(this);
            mView.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT));
            mView.setBackgroundColor(UIUtils.getColor(R.color.white));

            TextView tv = new TextView(this);
```

```
        FrameLayout.LayoutParams params = new
FrameLayout.LayoutParams(FrameLayout.LayoutParams.MATCH_PARENT, UIUtils.dip2Px(45));
        tv.setLayoutParams(params);
        tv.setGravity(Gravity.LEFT | Gravity.CENTER_VERTICAL);
        tv.setPadding(UIUtils.dip2Px(20), 0, 0, 0);
        tv.setTextColor(UIUtils.getColor(R.color.gray0));
        tv.setTextSize(14);
        tv.setText("");
        mView.addView(tv);

        tv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mPopupWindow.dismiss();
                //
                final String dirPath = new
File(Environment.getExternalStorageDirectory().getAbsolutePath(),
getPackageName()).getAbsolutePath();
                final String fileName = "header.jpg";
                OkHttpUtils.get().url(mUrl).build().execute(new FileCallBack(dirPath, fileName) {
                    @Override
                    public void onError(Call call, Exception e, int id) {
                        UIUtils.showToast("");
                    }

                    @Override
                    public void onResponse(File response, int id) {
                        UIUtils.showToast("" + dirPath + "/" + fileName);
                    }
                });
            }
        });
    }
    mPopupWindow = PopupWindowFactory.getPopupWindowAtLocation(mView,
ViewGroup.LayoutParams.MATCH_PARENT, ViewGroup.LayoutParams.WRAP_CONTENT,
getWindow().getDecorView().getRootView(), Gravity.BOTTOM, 0, 0);
    mPopupWindow.setOnDismissListener(new PopupWindow.OnDismissListener() {
        @Override
        public void onDismiss() {
            PopupWindowFactory.makeWindowLight(ShowBigImageActivity.this);
        }
    });
```

```
        PopupWindowFactory.makeWindowDark(this);
    }
}

39:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\SplashActivity.j
ava
package com.lqr.wechat.activity;

import android.content.Intent;
import android.text.TextUtils;
import android.view.Window;
import android.view.WindowManager;

import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimAccountSDK;
import com.lqr.wechat.utils.UIUtils;

import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class SplashActivity extends BaseActivity {


    @OnClick(R.id.btnLogin)
    public void login() {
        startActivity(new Intent(this, LoginActivity.class));
    }

    @OnClick(R.id.btnRegister)
    public void register() {
        UIUtils.showToast("");
//        startActivity(new Intent(this, RegisterActivity.class));
//        finish();
    }

    @Override
    public void init() {
        if (canAutoLogin()) {
```

```
            //
            startActivity(new Intent(this, MainActivity.class));
            finish();
        }
    }

    @Override
    public void initView() {
        requestWindowFeature(Window.FEATURE_NO_TITLE);//
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);//
        setContentView(R.layout.activity_splash);
        ButterKnife.inject(this);
    }

    /**
     *
     *
     * @return
     */
    public boolean canAutoLogin() {
        String account = NimAccountSDK.getUserAccount();
        String token = NimAccountSDK.getUserAccount();
        return !TextUtils.isEmpty(account) && !TextUtils.isEmpty(token);
    }

}
```

40:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\TeamAnnouncementEditActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.netease.nimlib.sdk.team.constant.TeamFieldEnum;
import com.netease.nimlib.sdk.team.model.Team;
```

```
import java.io.Serializable;
import java.util.HashMap;
import java.util.Map;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */

public class TeamAnnouncementEditActivity extends BaseActivity {

    public static final String TEAM = "team";

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.btnOk)
    Button mBtnOk;

    @InjectView(R.id.etContent)
    EditText mEtContent;
    private Team mTeam;

    @OnClick({R.id.btnOk})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.btnOk:
                final String content = mEtContent.getText().toString().trim();
                if (!TextUtils.isEmpty(content)) {
                    showMaterialDialog("", "?", "", "", new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            hideMaterialDialog();
                            Map<TeamFieldEnum, Serializable> fields = new HashMap<>(1);
                            fields.put(TeamFieldEnum.Announcement, content);
                            NimTeamSDK.updateTeamFields(mTeam.getId(), fields);
                            showWaitingDialog("");
                            //TODO:@
```

```
                    finish();
                }
            }, new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    hideMaterialDialog();
                }
            });

        }
        break;
    }
}

@Override
public void init() {
    mTeam = (Team) getIntent().getSerializableExtra(TEAM);
    if (mTeam == null) {
        interrupt();
    }
}

@Override
public void initView() {
    setContentView(R.layout.activity_team_announcement_edit);
    ButterKnife.inject(this);
    initToolbar();
}

private void initToolbar() {
    setSupportActionBar(mToolbar);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setTitle("");
    mToolbar.setNavigationIcon(R.mipmap.ic_back);
    mBtnOk.setVisibility(View.VISIBLE);
    mBtnOk.setText("");
}


}
```

41:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\TeamCheatCr

eateActvitiy.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.graphics.drawable.Drawable;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
import com.lqr.recyclerview.LQRRecyclerView;
import com.lqr.wechat.R;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.Contact;
import com.lqr.wechat.nimsdk.NimFriendSDK;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.SortUtils;
import com.lqr.wechat.utils.StringUtils;
import com.lqr.wechat.utils.UIUtils;
import com.lqr.wechat.view.QuickIndexBar;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.friend.model.Friend;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.team.constant.TeamFieldEnum;
import com.netease.nimlib.sdk.team.constant.TeamTypeEnum;
import com.netease.nimlib.sdk.team.model.Team;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
```

```
import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

import static com.lqr.wechat.R.id.ivHeader;

/**
 * @ CSDN_LQR
 * @
 */
public class TeamCheatCreateActvitiy extends BaseActivity {

    public static final String ADD_TEAM_MEMBER = "add_team_member";//
    private boolean isAddTeamMemberMode = false;//
    private List<String> mSelectedTeamMemberAccounts = new ArrayList<>();//

    private List<Contact> mContacts = new ArrayList<>();
    private List<Friend> mFriends = new ArrayList<>();
    private LQRAdapterForRecyclerView<Contact> mAdapter;
    private int i;

    private List<Contact> mSelectedContacts = new ArrayList<>();//
    private LQRAdapterForRecyclerView<Contact> mSelectedContactsAdapter;
    private Drawable mSearchDrawable;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.btnOk)
    Button mBtnOk;

    @InjectView(R.id.rvSelectedContacts)
    LQRRecyclerView mRvSelectedContacts;
    @InjectView(R.id.etKey)
    EditText mEtKey;
    @InjectView(R.id.vTop)
    View mVTop;

    @InjectView(R.id.rvContacts)
    LQRRecyclerView mRvContacts;
    @InjectView(R.id.quickIndexBar)
    QuickIndexBar mQuickIndexBar;
    @InjectView(R.id.tvLetter)
```

```
    TextView mTvLetter;

    private View mHeaderView;
    private TextView mTvSelectOneGroup;
    private TextView mTvCreateGroupFaceToFace;

    @OnClick({R.id.btnOk})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.btnOk:
                if (mSelectedContacts.size() == 0)
                    return;
                ArrayList<String> accounts = new ArrayList<>(mSelectedContacts.size());
                for (Contact contact : mSelectedContacts) {
                    accounts.add(contact.getAccount());
                }

                if (isAddTeamMemberMode) {
                    Intent intent = new Intent();
                    intent.putStringArrayListExtra(ADD_TEAM_MEMBER, accounts);
                    setResult(RESULT_OK, intent);
                    finish();
                } else {
                    showWaitingDialog("");
                    HashMap<TeamFieldEnum, Serializable> fields = new HashMap<>();
//                  fields.put(TeamFieldEnum.Name, "(" + accounts.size() + 1 + ")");
                    NimTeamSDK.createTeam(fields, TeamTypeEnum.Normal, accounts, new
RequestCallback<Team>() {
                        @Override
                        public void onSuccess(Team param) {
                            hideWaitingDialog();
                            //SessionActivity
                            Intent intent = new Intent(TeamCheatCreateActvitiy.this, SessionActivity.class);
                            intent.putExtra(SessionActivity.SESSION_ACCOUNT, param.getId());
                            intent.putExtra(SessionActivity.SESSION_TYPE, SessionTypeEnum.Team);
                            startActivity(intent);
                            setResult(RESULT_OK);
                            finish();
                        }

                        @Override
                        public void onFailed(int code) {
```

```
                    UIUtils.showToast("" + code);
                    hideWaitingDialog();
                }

                @Override
                public void onException(Throwable exception) {
                    exception.printStackTrace();
                    hideWaitingDialog();
                }
            });
        }
        break;
    }
  }

  @Override
  public void init() {
    //
    ArrayList<String> stringArrayListExtra =
getIntent().getStringArrayListExtra(ADD_TEAM_MEMBER);
    if (stringArrayListExtra == null) {
      isAddTeamMemberMode = false;
    } else {
      isAddTeamMemberMode = true;
    }
    if (!StringUtils.isEmpty(stringArrayListExtra)) {
      mSelectedTeamMemberAccounts.addAll(stringArrayListExtra);
    }
  }

  @Override
  public void initView() {
    setContentView(R.layout.activity_team_cheat_create);
    ButterKnife.inject(this);

    initToolbar();
    initHeaderView();

    //
    mSearchDrawable = UIUtils.getResource().getDrawable(R.mipmap.ic_search1);
    mSearchDrawable.setBounds(0, 0, mSearchDrawable.getMinimumWidth(),
mSearchDrawable.getMinimumHeight());
```

```
    }

    @Override
    public void initData() {
        try {
            mFriends.clear();
            mContacts.clear();

            //
            List<Friend> friends = NimFriendSDK.getFriends();
            if (!StringUtils.isEmpty(friends)) {
                mFriends.addAll(friends);

                //
                List<String> accountList = new ArrayList<>();
                for (int i = 0; i < mFriends.size(); i++) {
                    String account = mFriends.get(i).getAccount();
                    if (NimUserInfoSDK.getUser(account) == null) {
                        accountList.add(account);
                    }
                }

                //
                if (!StringUtils.isEmpty(accountList)) {
                    NimUserInfoSDK.getUserInfosFormServer(accountList, new
RequestCallback<List<NimUserInfo>>() {
                        @Override
                        public void onSuccess(List<NimUserInfo> param) {
                            setDataAndUpdateView();
                        }

                        @Override
                        public void onFailed(int code) {
                            UIUtils.showToast("" + code);
                        }

                        @Override
                        public void onException(Throwable exception) {
                            exception.printStackTrace();
                        }
                    });
                } else {
```

```
                setDataAndUpdateView();
            }
        } else {
            setDataAndUpdateView();
        }
    } catch (Exception e) {
        e.printStackTrace();
        initData();
    }

    setSelectedContactsAdapter();
}

@Override
public void initListener() {
    mQuickIndexBar.setListener(new QuickIndexBar.OnLetterUpdateListener() {
        @Override
        public void onLetterUpdate(String letter) {
            //
            showLetter(letter);

            //
            if ("".equalsIgnoreCase(letter)) {
                mRvContacts.moveToPosition(0);
            } else if ("".equalsIgnoreCase(letter)) {
                mRvContacts.moveToPosition(0);
            } else {
                //
                for (i = 0; i < mContacts.size(); i++) {
                    Contact contact = mContacts.get(i);
                    String c = contact.getPinyin().charAt(0) + "";
                    if (c.equalsIgnoreCase(letter)) {
                        mRvContacts.moveToPosition(i);
                        break;
                    }
                }
            }
        }
    });
}

@Override
```

```
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
        mBtnOk.setVisibility(View.VISIBLE);
        mBtnOk.setText("");
    }

    private void initHeaderView() {
        mHeaderView = View.inflate(this, R.layout.header_group_cheat_rv, null);
        mTvSelectOneGroup = (TextView) mHeaderView.findViewById(R.id.tvSelectOneGroup);
        mTvCreateGroupFaceToFace = (TextView)
mHeaderView.findViewById(R.id.tvCreateGroupFaceToFace);
    }

    private void setDataAndUpdateView() {
        if (mFriends != null) {
            for (int i = 0; i < mFriends.size(); i++) {
                mContacts.add(new Contact(mFriends.get(i).getAccount()));
            }
            //
            SortUtils.sortContacts(mContacts);
        }
        setContactsAdapter();
    }

    /**
     *
     */
    private void setContactsAdapter() {
        mAdapter = new LQRAdapterForRecyclerView<Contact>(this, R.layout.item_contact_cv,
mContacts) {
            @Override
```

```
        public void convert(final LQRViewHolderForRecyclerView helper, final Contact item, int position) {
            helper.setText(R.id.tvName, TextUtils.isEmpty(item.getAlias()) ? item.getName() : item.getAlias());
            if (!TextUtils.isEmpty(item.getAvatar())) {
                ImageLoaderManager.LoadNetImage(item.getAvatar(), (ImageView) helper.getView(ivHeader));
            } else {
                helper.setImageResource(ivHeader, R.mipmap.default_header);
            }

            String str = "";
            //
            String currentLetter = item.getPinyin().charAt(0) + "";

            if (position == 0) {
                str = currentLetter;
            } else {
                //
                String preLetter = mContacts.get(position - 1).getPinyin().charAt(0) + "";
                //
                if (!preLetter.equalsIgnoreCase(currentLetter)) {
                    str = currentLetter;
                }

                int nextIndex = position + 1;
                if (nextIndex < mContacts.size() - 1) {
                    //
                    String nextLetter = mContacts.get(nextIndex).getPinyin().charAt(0) + "";
                    //
                    if (!nextLetter.equalsIgnoreCase(currentLetter)) {
                        helper.setViewVisibility(R.id.vLine, View.INVISIBLE);
                    } else {
                        helper.setViewVisibility(R.id.vLine, View.VISIBLE);
                    }
                } else {
                    helper.setViewVisibility(R.id.vLine, View.INVISIBLE);
                }
            }
            if (position == mContacts.size() - 1) {
                helper.setViewVisibility(R.id.vLine, View.GONE);
            }
```

```
//str
if (TextUtils.isEmpty(str)) {
    helper.setViewVisibility(R.id.tvIndex, View.GONE);
} else {
    helper.setViewVisibility(R.id.tvIndex, View.VISIBLE);
    helper.setText(R.id.tvIndex, currentLetter);
}


final CheckBox cb = helper.getView(R.id.cb);
helper.setViewVisibility(R.id.cb, View.VISIBLE);

if (isAddTeamMemberMode) {
    //
    if (mSelectedTeamMemberAccounts.contains(item.getAccount())) {
        cb.setEnabled(false);
        cb.setChecked(true);
    } else {
        cb.setEnabled(true);
    }
}

//
helper.getView(R.id.root).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (isAddTeamMemberMode) {
            //
            if (mSelectedTeamMemberAccounts.contains(item.getAccount())) {
                return;
            }
        }

        if (cb.isChecked()) {
            cb.setChecked(false);
            //
            mSelectedContactsAdapter.removeItem(item);
        } else {
            cb.setChecked(true);
            //
            mSelectedContactsAdapter.addLastItem(item);
```

```
                }
                mBtnOk.setText("" + (mSelectedContacts.size() > 0 ? "(" +
mSelectedContacts.size() + ")" : ""));
                //
                if (mSelectedContacts.size() > 0) {
                    mEtKey.setCompoundDrawables(null, null, null, null);
                } else {
                    mEtKey.setCompoundDrawables(mSearchDrawable, null, null, null);
                }
            }
        });

    }
};
//
mAdapter.addHeaderView(mHeaderView);
//
if (mRvContacts != null)
    mRvContacts.setAdapter(mAdapter.getHeaderAndFooterAdapter());
}

/**
 *
 */
private void setSelectedContactsAdapter() {
//    for (int i = 0; i < 10; i++) {
//        mSelectedContacts.add(new Contact());
//    }
    if (mSelectedContactsAdapter == null) {
        mSelectedContactsAdapter = new LQRAdapterForRecyclerView<Contact>(this,
R.layout.item_selected_contact_rv, mSelectedContacts) {
            @Override
            public void convert(LQRViewHolderForRecyclerView helper, Contact item, int position) {

                //
                LinearLayout.LayoutParams params = (LinearLayout.LayoutParams)
mRvSelectedContacts.getLayoutParams();
//                params.weight = mSelectedContacts.size() > 5 ? 4 : 0;
                int parentWidth = ((LinearLayout) mRvSelectedContacts.getParent()).getWidth();
                int childWidth = parentWidth * 4 / 5;
                params.width = mSelectedContacts.size() > 5 ? childWidth :
params.WRAP_CONTENT;
```

```
                mRvSelectedContacts.setLayoutParams(params);

                ImageView ivHeader = helper.getView(R.id.ivHeader);
                if (TextUtils.isEmpty(item.getAvatar())) {
                    ivHeader.setImageResource(R.mipmap.default_header);
                } else {
                    ImageLoaderManager.LoadNetImage(item.getAvatar(), ivHeader);
                }
            }
        };
        mRvSelectedContacts.setAdapter(mSelectedContactsAdapter);
    } else {
        mSelectedContactsAdapter.notifyDataSetChanged();
    }
}

/**
 *
 *
 * @param letter
 */
protected void showLetter(String letter) {
    mTvLetter.setVisibility(View.VISIBLE);//
    mTvLetter.setText(letter);

    UIUtils.getMainThreadHandler().removeCallbacksAndMessages(null);
    UIUtils.postTaskDelay(new Runnable() {
        @Override
        public void run() {
            mTvLetter.setVisibility(View.GONE);
        }
    }, 500);
}
}
```

42:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\TeamCheatInfoActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
```

```
import android.view.MenuItem;
import android.view.View;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.kyleduo.switchbutton.SwitchButton;
import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
import com.lqr.optionitemview.OptionItemView;
import com.lqr.recyclerview.LQRRecyclerView;
import com.lqr.wechat.R;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.UserCache;
import com.lqr.wechat.nimsdk.NimHistorySDK;
import com.lqr.wechat.nimsdk.NimRecentContactSDK;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.StringUtils;
import com.lqr.wechat.utils.UIUtils;
import com.lqr.wechat.view.CustomDialog;
import com.netease.nimlib.sdk.InvocationFuture;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.team.constant.TeamMemberType;
import com.netease.nimlib.sdk.team.model.Team;
import com.netease.nimlib.sdk.team.model.TeamMember;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
```

```
 */
public class TeamCheatInfoActivity extends BaseActivity {

    public static final String GROUP_CHEAT_INFO_TEAMID = "teamId";
    public static final int REQ_ADD_MEMBERS = 1000;
    public static final int REQ_REMOVE_MEMBERS = 1001;
    public static final int REQ_CHANGE_NAME = 1002;
    public static final int REQ_WATCH_USER_INFO = 1003;
    public static final int RESP_QUIT_TEAM = 10000;
    public static final int RESP_CHEAT_SINGLE = 10001;
    public static final int RESP_CLEAR_CHATTING_RECORD_HISTORY = 10002;

    private Intent mIntent;
    private String mTeamId;
    private Team mTeam;
    private boolean mIsClearChattingHistory = false;
    private boolean mIsManager;//

    private Observer<TeamMember> memberRemoveObserver;
    private Observer<List<TeamMember>> memberUpdateObserver;

    private List<TeamMember> mTeamMemberList = new ArrayList<>();
    private LQRAdapterForRecyclerView mAdapter;
    private CustomDialog mDialog;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.rvMember)
    LQRRecyclerView mRvMember;
    @InjectView(R.id.oivTeamName)
    OptionItemView mOivTeamName;
    @InjectView(R.id.tvAnnouncement)
    TextView mTvAnnouncement;
    @InjectView(R.id.vLineTeamManage)
    View mVLineTeamManage;
    @InjectView(R.id.oivTeamManage)
    OptionItemView mOivTeamManage;
    @InjectView(R.id.oivNickNameInTeam)
    OptionItemView mOivNickNameInTeam;
    @InjectView(R.id.llShowNickName)
    LinearLayout mLlShowNickName;
    @InjectView(R.id.sbShowNickName)
```

```
    SwitchButton mSbShowNickName;

    @OnClick({R.id.oivTeamName, R.id.oivQRCordCard, R.id.llAnnouncement,
R.id.oivNickNameInTeam, R.id.btnQuitTeam, R.id.oivClearMsgRecord})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.oivTeamName:
                mIntent = new Intent(this, TeamNameSetActivity.class);
                mIntent.putExtra(TeamNameSetActivity.TEAM_ID, mTeamId);
                startActivity(mIntent);
                break;
            case R.id.oivQRCordCard:
                mIntent = new Intent(this, QRCodeCardActivity.class);
                mIntent.putExtra(QRCodeCardActivity.QRCODE_TEAM, mTeam);
                startActivity(mIntent);
                break;
            case R.id.llAnnouncement:
                if (mIsManager) {
                    //
                    mIntent = new Intent(this, TeamAnnouncementEditActivity.class);
                    mIntent.putExtra(TeamAnnouncementEditActivity.TEAM, mTeam);
                    startActivity(mIntent);
                } else {
                    showMaterialDialog("", "", "", "", new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            hideMaterialDialog();
                        }
                    }, null);
                }
                break;
            case R.id.oivNickNameInTeam:
                showChangeNickNameDialog();
                break;
            case R.id.btnQuitTeam:
                showWaitingDialog("");
                NimTeamSDK.quitTeam(mTeamId, new RequestCallback<Void>() {
                    @Override
                    public void onSuccess(Void param) {
                        hideWaitingDialog();
                        //
                        NimRecentContactSDK.deleteRecentContactAndNotify(mTeamId,
```

```
SessionTypeEnum.Team);
                    setResult(RESP_QUIT_TEAM);
                    onBackPressed();
                }

                @Override
                public void onFailed(int code) {
                    hideWaitingDialog();
                    UIUtils.showToast("" + code);
                }

                @Override
                public void onException(Throwable exception) {
                    hideWaitingDialog();
                    UIUtils.showToast("");
                    exception.printStackTrace();
                }
            });
            break;
        case R.id.oivClearMsgRecord:
            showMaterialDialog("", "?", "", "", new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    hideMaterialDialog();
                    NimHistorySDK.clearChattingHistory(mTeamId, SessionTypeEnum.Team);
                    mIsClearChattingHistory = true;
                }
            }, new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    hideMaterialDialog();
                }
            });
            break;
    }
}

@Override
public void init() {
    mTeamId =
getIntent().getStringExtra(TeamCheatInfoActivity.GROUP_CHEAT_INFO_TEAMID);
    if (TextUtils.isEmpty(mTeamId)) {
```

```
            interrupt();
            return;
        }

        mTeam = NimTeamSDK.queryTeamBlock(mTeamId);
//        mIsManager = UserCache.getAccount().equals(mTeam.getCreator());
        TeamMemberType myMemberType = NimTeamSDK.queryTeamMemberBlock(mTeamId,
UserCache.getAccount()).getType();
        if (myMemberType == TeamMemberType.Manager || myMemberType ==
TeamMemberType.Owner) {
            mIsManager = true;
        } else {
            mIsManager = false;
        }

        //
        observeMemberUpdate();
        observeMemberRemove();
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_team_cheat_info);
        ButterKnife.inject(this);

        initToolbar();
    }

    @Override
    protected void onResume() {
        super.onResume();
        mTeam = NimTeamSDK.queryTeamBlock(mTeamId);
        getSupportActionBar().setTitle("(" + mTeam.getMemberCount() + ")");
        mOivTeamName.setRightText(TextUtils.isEmpty(mTeam.getName()) ? "" :
mTeam.getName());
mOivNickNameInTeam.setRightText(NimTeamSDK.getTeamMemberDisplayNameWithoutMe(mT
eamId, UserCache.getAccount()));
        if (!TextUtils.isEmpty(mTeam.getAnnouncement())) {
            mTvAnnouncement.setVisibility(View.VISIBLE);
            mTvAnnouncement.setText(mTeam.getAnnouncement());
        } else {
            mTvAnnouncement.setVisibility(View.GONE);
```

```
        }

//        if (mIsManager) {
//            mVLineTeamManage.setVisibility(View.VISIBLE);
//            mOivTeamManage.setVisibility(View.VISIBLE);
//        } else {
//            mVLineTeamManage.setVisibility(View.GONE);
//            mOivTeamManage.setVisibility(View.GONE);
//        }

        //
        mSbShowNickName.setChecked(NimTeamSDK.shouldShowNickName(mTeamId));
    }

    @Override
    public void initData() {
        //
        NimTeamSDK.queryMemberList(mTeamId, new RequestCallback<List<TeamMember>>() {
            @Override
            public void onSuccess(List<TeamMember> param) {
                if (!StringUtils.isEmpty(param)) {
                    mTeamMemberList.clear();
                    mTeamMemberList.addAll(param);
                    if (mIsManager) {
                        mTeamMemberList.add(null);
                        mTeamMemberList.add(null);
                    } else {
                        mTeamMemberList.add(null);
                    }
                    //
                    List<String> accountList = new ArrayList<>(param.size());
                    for (TeamMember tm : param) {
                        accountList.add(tm.getAccount());
                    }
                    if (!StringUtils.isEmpty(accountList)) {
                        NimUserInfoSDK.getUserInfosFormServer(accountList, new
RequestCallback<List<NimUserInfo>>() {
                            @Override
                            public void onSuccess(List<NimUserInfo> param) {
                                setAdapter();
                            }
```

```
                    @Override
                    public void onFailed(int code) {
                        UIUtils.showToast("" + code);
                    }

                    @Override
                    public void onException(Throwable exception) {
                        exception.printStackTrace();
                    }
                });
            }
        }
    }

    @Override
    public void onFailed(int code) {
        UIUtils.showToast("" + code);
    }

    @Override
    public void onException(Throwable exception) {
        exception.printStackTrace();
    }
  });
}

@Override
public void initListener() {
  mLlShowNickName.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      mSbShowNickName.setChecked(!mSbShowNickName.isChecked());
    }
  });
  mSbShowNickName.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
      showWaitingDialog("");
      NimTeamSDK.setShouldShowNickName(mTeamId, isChecked, new
RequestCallback<Void>() {
        @Override
```

```
                public void onSuccess(Void param) {
                    hideWaitingDialog();
                }

                @Override
                public void onFailed(int code) {
                    UIUtils.showToast("" + code);
                    hideWaitingDialog();
                }

                @Override
                public void onException(Throwable exception) {
                    exception.printStackTrace();
                    hideWaitingDialog();
                }
            });
        }
    });
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            onBackPressed();
            break;
    }
    return super.onOptionsItemSelected(item);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode) {
        case REQ_ADD_MEMBERS:
            if (resultCode == RESULT_OK) {
                showWaitingDialog("");
                //
                ArrayList<String> accounts =
data.getStringArrayListExtra(TeamCheatCreateActvitiy.ADD_TEAM_MEMBER);
                NimTeamSDK.addMembers(mTeamId, accounts, new RequestCallback<Void>() {
                    @Override
                    public void onSuccess(Void param) {
```

```
                hideWaitingDialog();
            }

            @Override
            public void onFailed(int code) {
                UIUtils.showToast("" + code);
                hideWaitingDialog();
            }

            @Override
            public void onException(Throwable exception) {
                exception.printStackTrace();
                UIUtils.showToast("");
                hideWaitingDialog();
            }
        });
    }
    break;
case REQ_REMOVE_MEMBERS:
    if (resultCode == RESULT_OK) {
        showWaitingDialog("");
        ArrayList<String> accounts =
data.getStringArrayListExtra(TeamCheatRemoveMemberActivity.REMOVE_TEAM_MEMBER);
        InvocationFuture<Void> invocationFuture =
NimTeamSDK.removeMembers(mTeamId, accounts);
        invocationFuture.setCallback(new RequestCallback<Void>() {
            @Override
            public void onSuccess(Void param) {
                hideWaitingDialog();
            }

            @Override
            public void onFailed(int code) {
                UIUtils.showToast("" + code);
                hideWaitingDialog();
            }

            @Override
            public void onException(Throwable exception) {
                exception.printStackTrace();
                UIUtils.showToast("");
                hideWaitingDialog();
```

```
                }
            });
        }
        break;
    case REQ_WATCH_USER_INFO:
        if (resultCode == RESULT_OK) {
            setResult(RESP_CHEAT_SINGLE);
            onBackPressed();
        }
        break;
  }
  super.onActivityResult(requestCode, resultCode, data);
}

@Override
protected void onDestroy() {
  super.onDestroy();
  NimTeamSDK.observeMemberUpdate(memberUpdateObserver, false);
  NimTeamSDK.observeMemberRemove(memberRemoveObserver, false);
}

@Override
public void onBackPressed() {
  if (mIsClearChattingHistory)
    setResult(RESP_CLEAR_CHATTING_RECORD_HISTORY);
  super.onBackPressed();
}

private void initToolbar() {
  setSupportActionBar(mToolbar);
  getSupportActionBar().setDisplayHomeAsUpEnabled(true);
  mToolbar.setNavigationIcon(R.mipmap.ic_back);
}

private void setAdapter() {
  if (mAdapter == null) {
    mAdapter = new LQRAdapterForRecyclerView(this,
R.layout.item_member_info_group_cheat_rv, mTeamMemberList) {
        @Override
        public void convert(LQRViewHolderForRecyclerView helper, Object obj, int position) {
            final ImageView ivHeader = helper.getView(R.id.ivHeader);
            if (mIsManager && position >= mTeamMemberList.size() - 2) {//+-
```

```
        if (position == mTeamMemberList.size() - 2) {//+
            ivHeader.setImageResource(R.mipmap.ic_add_team_member);
            helper.getView(R.id.root).setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    //
                    addMembers();
                }
            });
        } else {//-
            helper.getView(R.id.root).setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    //
                    removeMember();
                }
            });
            ivHeader.setImageResource(R.mipmap.ic_remove_team_member);
        }
        helper.setText(R.id.tvName, "");
    } else if (!mIsManager && position >= mTeamMemberList.size() - 1) {//+
        helper.getView(R.id.root).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //
                addMembers();
            }
        });
        ivHeader.setImageResource(R.mipmap.ic_add_team_member);
        helper.setText(R.id.tvName, "");
    } else {
        final TeamMember item = (TeamMember) obj;
        helper.getView(R.id.root).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //
                Intent intent = new Intent(TeamCheatInfoActivity.this, UserInfoActivity.class);
                intent.putExtra(UserInfoActivity.USER_INFO_ACCOUNT, item.getAccount());
                startActivityForResult(intent, REQ_WATCH_USER_INFO);
            }
        });
        helper.setText(R.id.tvName,
```

```
NimTeamSDK.getTeamMemberDisplayNameWithoutMe(item.getTid(), item.getAccount()));
                String account = item.getAccount();
                NimUserInfo userInfo = NimUserInfoSDK.getUser(account);
                if (userInfo == null) {
                    NimUserInfoSDK.getUserInfoFromServer(account, new
RequestCallback<List<NimUserInfo>>() {
                        @Override
                        public void onSuccess(List<NimUserInfo> param) {
                            if (!StringUtils.isEmpty(param)) {
                                NimUserInfo userInfo = param.get(0);
                                if (!TextUtils.isEmpty(userInfo.getAvatar())) {
                                    ImageLoaderManager.LoadNetImage(userInfo.getAvatar(), ivHeader);
                                } else {
                                    ivHeader.setImageResource(R.mipmap.default_header);
                                }
                            }
                        }

                        @Override
                        public void onFailed(int code) {
                            ivHeader.setImageResource(R.mipmap.default_header);
                        }

                        @Override
                        public void onException(Throwable exception) {
                            ivHeader.setImageResource(R.mipmap.default_header);
                        }
                    });
                } else {
                    if (!TextUtils.isEmpty(userInfo.getAvatar())) {
                        ImageLoaderManager.LoadNetImage(userInfo.getAvatar(), ivHeader);
                    } else {
                        ivHeader.setImageResource(R.mipmap.default_header);
                    }
                }
            }
        }
    };
    mRvMember.setAdapter(mAdapter);
} else {
    mAdapter.notifyDataSetChanged();
}
```

```
    }

    /**
     *
     */
    private void addMembers() {
        //
        ArrayList<String> selectedTeamMemberAccounts = new ArrayList<>();
        for (int i = 0; i < mTeam.getMemberCount(); i++) {
            selectedTeamMemberAccounts.add(mTeamMemberList.get(i).getAccount());
        }

        Intent intent = new Intent(this, TeamCheatCreateActvitiy.class);
        intent.putStringArrayListExtra(TeamCheatCreateActvitiy.ADD_TEAM_MEMBER,
selectedTeamMemberAccounts);
        startActivityForResult(intent, REQ_ADD_MEMBERS);
    }

    /**
     *
     */
    private void removeMember() {
        Intent intent = new Intent(this, TeamCheatRemoveMemberActivity.class);
        intent.putExtra(TeamCheatRemoveMemberActivity.TEAMID, mTeamId);
        startActivityForResult(intent, REQ_REMOVE_MEMBERS);
    }

    private void observeMemberUpdate() {
        memberUpdateObserver = new Observer<List<TeamMember>>() {
            @Override
            public void onEvent(List<TeamMember> teamMembers) {
                initData();
                onResume();
            }
        };
        NimTeamSDK.observeMemberUpdate(memberUpdateObserver, true);
    }

    private void observeMemberRemove() {
        memberRemoveObserver = new Observer<TeamMember>() {
            @Override
            public void onEvent(TeamMember teamMember) {
```

```
                initData();
                onResume();
            }
        };
        NimTeamSDK.observeMemberRemove(memberRemoveObserver, true);
    }

    private void showChangeNickNameDialog() {
        View view = View.inflate(this, R.layout.dialog_team_nick_change, null);
        mDialog = new CustomDialog(this, view, R.style.dialog);
        mDialog.setCancelable(false);
        mDialog.show();
        final EditText etName = (EditText) view.findViewById(R.id.etName);
//        String nickName = NimTeamSDK.getTeamNick(mTeamId, UserCache.getAccount());
        String nickName = NimTeamSDK.getTeamMemberDisplayNameWithoutMe(mTeamId,
UserCache.getAccount());
        etName.setText(nickName);
        if (!TextUtils.isEmpty(nickName) && nickName.length() > 0)
            etName.setSelection(nickName.length());
        view.findViewById(R.id.tvCancle).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mDialog.dismiss();
                mDialog = null;
            }
        });
        view.findViewById(R.id.tvOk).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //
                String newNickName = etName.getText().toString().trim();
                if (!TextUtils.isEmpty(newNickName)) {
                    NimTeamSDK.updateMyTeamNick(mTeamId, newNickName, new
RequestCallback<Void>() {
                        @Override
                        public void onSuccess(Void param) {
                            UIUtils.showToast("");
                            mDialog.dismiss();
                            mDialog = null;
                            onResume();
                        }
```

```
                @Override
                public void onFailed(int code) {
                    switch (code) {
                        case 805:
                            UIUtils.showToast("");
                            break;
                        default:
                            UIUtils.showToast("" + code);
                            break;
                    }
                }

                @Override
                public void onException(Throwable exception) {
                    UIUtils.showToast("");
                    exception.printStackTrace();
                }
            });
        }
    }
});
    }
}
```

43:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\TeamCheatListActivity.java

```
package com.lqr.wechat.activity;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
```

```
import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
import com.lqr.ninegridimageview.LQRNineGridImageView;
import com.lqr.ninegridimageview.LQRNineGridImageViewAdapter;
import com.lqr.recyclerview.LQRRecyclerView;
import com.lqr.wechat.R;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.ResponseCode;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.team.model.Team;
import com.netease.nimlib.sdk.team.model.TeamMember;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class TeamCheatListActivity extends BaseActivity {

    private List<Team> mMyTeamList = new ArrayList<>();
    private LQRAdapterForRecyclerView<Team> mAdapter;
    private TextView mHeaderView;
    private TextView mFooterTv;
    private LQRNineGridImageViewAdapter<NimUserInfo> mNineGridAdapter;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.llContent)
    LinearLayout mLlContent;
```

```
    @InjectView(R.id.tvTip)
    TextView mTvTip;
    @InjectView(R.id.rvTeamList)
    LQRRecyclerView mRvTeamList;

    @Override
    public void initView() {
        setContentView(R.layout.activity_team_cheat_list);
        ButterKnife.inject(this);
        initToolbar();
        initHeaderViewAndFooterView();
        mNineGridAdapter = new LQRNineGridImageViewAdapter<NimUserInfo>() {
            @Override
            protected void onDisplayImage(Context context, ImageView imageView, NimUserInfo
userInfo) {
                if (!TextUtils.isEmpty(userInfo.getAvatar())) {
                    ImageLoaderManager.LoadNetImage(userInfo.getAvatar(), imageView);
                } else {
                    imageView.setImageResource(R.mipmap.default_header);
                }
            }
        };
    }

    @Override
    public void initData() {
        NimTeamSDK.queryTeamList(new RequestCallbackWrapper<List<Team>>() {
            @Override
            public void onResult(int code, List<Team> result, Throwable exception) {
                if (code == ResponseCode.RES_SUCCESS && result != null && exception == null) {
                    mLlContent.setVisibility(View.VISIBLE);
                    mTvTip.setVisibility(View.GONE);

                    mMyTeamList.clear();
                    mMyTeamList.addAll(result);

                    setAdapter();

                } else {
                    mLlContent.setVisibility(View.GONE);
                    mTvTip.setVisibility(View.VISIBLE);
                    exception.printStackTrace();
```

```
            }
        }
    });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    new MenuInflater(this).inflate(R.menu.menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            break;
        case R.id.itemSearch:
            break;
        case R.id.itemMore:
            Intent intent = new Intent(this, TeamCheatCreateActvitiy.class);
            startActivityForResult(intent, 100);
            break;
    }
    return super.onOptionsItemSelected(item);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK) {
        finish();
    }
    super.onActivityResult(requestCode, resultCode, data);
}

private void initToolbar() {
    setSupportActionBar(mToolbar);
    getSupportActionBar().setTitle("");
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    mToolbar.setNavigationIcon(R.mipmap.ic_back);
}
```

```
    private void initHeaderViewAndFooterView() {
        mHeaderView = new TextView(this);
        ViewGroup.LayoutParams params1 = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, UIUtils.dip2Px(23));
        mHeaderView.setBackgroundColor(Color.parseColor("#E5E5E5"));
        mHeaderView.setGravity(Gravity.CENTER_VERTICAL);
        mHeaderView.setPadding(UIUtils.dip2Px(15), 0, 0, 0);
        mHeaderView.setText("");
        mHeaderView.setTextColor(Color.parseColor("#989898"));
        mHeaderView.setTextSize(13);
        mHeaderView.setLayoutParams(params1);

        mFooterTv = new TextView(this);
        ViewGroup.LayoutParams params2 = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, UIUtils.dip2Px(50));
        mFooterTv.setLayoutParams(params2);
        mFooterTv.setGravity(Gravity.CENTER);
    }

    private void setAdapter() {
        if (mAdapter == null) {
            mAdapter = new LQRAdapterForRecyclerView<Team>(this, R.layout.item_contact_cv,
mMyTeamList) {
                @Override
                public void convert(final LQRViewHolderForRecyclerView helper, final Team item, int
position) {
                    helper.setViewVisibility(R.id.ivHeader, View.GONE)
                            .setViewVisibility(R.id.ngiv, View.GONE);
                    final LQRNineGridImageView ngivHeader = helper.getView(R.id.ngiv);
                    NimTeamSDK.queryMemberList(item.getId(), new
RequestCallback<List<TeamMember>>() {
                        @Override
                        public void onSuccess(List<TeamMember> memberList) {
                            //
                            if (!TextUtils.isEmpty(item.getName()))
                                helper.setText(R.id.tvName, item.getName());
                            else {
                                StringBuilder sb = new StringBuilder();
                                for (int i = 0; i < memberList.size(); i++) {
                                    TeamMember member = memberList.get(i);
sb.append(NimTeamSDK.getTeamMemberDisplayNameWithYou(item.getId(), member.getAccou
nt()));
```

```
                    if (i != memberList.size() - 1) {
                        sb.append("");
                    }
                }
                helper.setText(R.id.tvName, sb.toString());
            }

            //
            if (memberList != null && memberList.size() > 0) {
                List<String> accounts = new ArrayList<>();
                int count = memberList.size() > 9 ? 9 : memberList.size();
                for (int i = 0; i < count; i++) {
                    accounts.add(memberList.get(i).getAccount());
                }
                NimUserInfoSDK.getUserInfosFormServer(accounts, new
RequestCallback<List<NimUserInfo>>() {
                    @Override
                    public void onSuccess(List<NimUserInfo> result) {
                        ngivHeader.setAdapter(mNineGridAdapter);
                        ngivHeader.setImagesData(result);
                    }

                    @Override
                    public void onFailed(int code) {

                    }

                    @Override
                    public void onException(Throwable exception) {

                    }
                });
            }
        }

        @Override
        public void onFailed(int code) {

        }

        @Override
        public void onException(Throwable exception) {
```

```
                    exception.printStackTrace();
                }
            }

        );


        //
        helper.getView(R.id.root).
                setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        //SessionActivity
                        Intent intent = new Intent(TeamCheatListActivity.this,
SessionActivity.class);
                        intent.putExtra(SessionActivity.SESSION_ACCOUNT,
item.getId());
                        intent.putExtra(SessionActivity.SESSION_TYPE,
SessionTypeEnum.Team);
                        startActivity(intent);
                        finish();
                    }
                }

        );
    }
}

;

mAdapter.addHeaderView(mHeaderView);
mFooterTv.setText(mMyTeamList.size() + "");
mAdapter.addFooterView(mFooterTv);
mRvTeamList.setAdapter(mAdapter.getHeaderAndFooterAdapter());
} else {
mAdapter.notifyDataSetChanged();
mAdapter.getHeaderAndFooterAdapter().notifyDataSetChanged();
}
}

}
```

44:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\TeamCheatRemoveMemberActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;

import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
import com.lqr.recyclerview.LQRRecyclerView;
import com.lqr.wechat.R;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.UserCache;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.team.model.Team;
import com.netease.nimlib.sdk.team.model.TeamMember;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class TeamCheatRemoveMemberActivity extends BaseActivity {

    public static final String TEAMID = "teamId";
    public static final String REMOVE_TEAM_MEMBER = "remove_team_member";//
```

```
    private String mTeamId;
    private Team mTeam;
    private List<TeamMember> mTeamMembers = new ArrayList<>();
    private ArrayList<String> mWillBeRemovedAccounts = new ArrayList<>();
    private LQRAdapterForRecyclerView<TeamMember> mAdapter;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.btnOk)
    Button mBtnOk;

    @InjectView(R.id.etKey)
    EditText mEtKey;
    @InjectView(R.id.rvMember)
    LQRRecyclerView mRvMember;

    @OnClick({R.id.btnOk})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.btnOk:
                if (mWillBeRemovedAccounts.size() > 0) {
                    Intent intent = new Intent();
                    intent.putStringArrayListExtra(REMOVE_TEAM_MEMBER,
mWillBeRemovedAccounts);
                    setResult(RESULT_OK, intent);
                    finish();
                }
                break;
        }
    }

    @Override
    public void init() {
        mTeamId = getIntent().getStringExtra(TEAMID);
        if (TextUtils.isEmpty(mTeamId)) {
            interrupt();
        }

        mTeam = NimTeamSDK.queryTeamBlock(mTeamId);
    }

    @Override
```

```
public void initView() {
    setContentView(R.layout.activity_team_cheat_remove_member);
    ButterKnife.inject(this);
    initToolbar();
}

@Override
public void initData() {
    //
    NimTeamSDK.queryMemberList(mTeamId, new RequestCallback<List<TeamMember>>() {
        @Override
        public void onSuccess(List<TeamMember> param) {
            mTeamMembers.clear();
            mTeamMembers.addAll(param);

            //
            int creatorPosi = -1;
            for (int i = 0; i < param.size(); i++) {
                TeamMember tm = param.get(i);
                if (mTeam.getCreator().equals(tm.getAccount())) {
                    creatorPosi = i;
                    break;
                }
            }
            if (creatorPosi != -1) {
                mTeamMembers.remove(creatorPosi);
                mTeamMembers.add(0, param.get(creatorPosi));
            }
            setAdapter();
        }

        @Override
        public void onFailed(int code) {
            UIUtils.showToast("" + code);
        }

        @Override
        public void onException(Throwable exception) {
            exception.printStackTrace();
        }
    });
}
```

```
    private void setAdapter() {
        if (mAdapter == null) {
            mAdapter = new LQRAdapterForRecyclerView<TeamMember>(this,
R.layout.item_contact_cv, mTeamMembers) {
                @Override
                public void convert(LQRViewHolderForRecyclerView helper, final TeamMember item, int
position) {
                    helper.setText(R.id.tvName,
NimTeamSDK.getTeamMemberDisplayNameWithoutMe(item.getTid(), item.getAccount()));
                    ImageView ivHeader = helper.getView(R.id.ivHeader);
                    NimUserInfo userInfo = NimUserInfoSDK.getUser(item.getAccount());
                    if (userInfo != null && !TextUtils.isEmpty(userInfo.getAvatar())) {
                        ImageLoaderManager.LoadNetImage(userInfo.getAvatar(), ivHeader);
                    } else {
                        ivHeader.setImageResource(R.mipmap.default_header);
                    }
                    final CheckBox cb = helper.getView(R.id.cb);
                    if (UserCache.getAccount().equals(item.getAccount())) {
                        cb.setVisibility(View.GONE);
                    } else {
                        cb.setVisibility(View.VISIBLE);
                    }

                    helper.getView(R.id.root).setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            if (UserCache.getAccount().equals(item.getAccount())) {
                                return;
                            } else {
                                if (cb.isChecked()) {
                                    cb.setChecked(false);
                                    mWillBeRemovedAccounts.remove(item.getAccount());
                                } else {
                                    cb.setChecked(true);
                                    mWillBeRemovedAccounts.add(item.getAccount());
                                }
                            }
                        }
                    });
                }
            };
```

```
            mRvMember.setAdapter(mAdapter);
        } else {
            mAdapter.notifyDataSetChanged();
        }
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("(" + mTeam.getMemberCount() + ")");
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
        mBtnOk.setVisibility(View.VISIBLE);
        mBtnOk.setText("");
        mBtnOk.setBackgroundResource(R.drawable.shape_btn_delete);
    }
}
```

45:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\TeamNameSetActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.lqr.wechat.R;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.InvocationFuture;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.team.constant.TeamFieldEnum;
import com.netease.nimlib.sdk.team.model.Team;

import java.io.Serializable;
import java.util.HashMap;
import java.util.Map;

import butterknife.ButterKnife;
import butterknife.InjectView;
```

```
import butterknife.OnClick;


/**
 * @ CSDN_LQR
 * @
 */
public class TeamNameSetActivity extends BaseActivity {

    public static final String TEAM_ID = "teamId";

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.btnOk)
    Button mBtnOk;

    @InjectView(R.id.etName)
    EditText mEtName;
    private String mTeamId;
    private Team mTeam;

    @OnClick(R.id.btnOk)
    public void click() {
        final String teamName = mEtName.getText().toString().trim();
//        if (!TextUtils.isEmpty(teamName)) {
            showWaitingDialog("");
            Map<TeamFieldEnum, Serializable> fields = new HashMap<>(1);
            fields.put(TeamFieldEnum.Name, teamName);
            InvocationFuture<Void> invocationFuture = NimTeamSDK.updateTeamFields(mTeamId,
fields);
            invocationFuture.setCallback(new RequestCallback<Void>() {
                @Override
                public void onSuccess(Void param) {
                    hideWaitingDialog();
                    finish();
                }

                @Override
                public void onFailed(int code) {
                    UIUtils.showToast("" + code);
                    hideWaitingDialog();
                }
```

```
            @Override
            public void onException(Throwable exception) {
                exception.printStackTrace();
                hideWaitingDialog();
            }
        });
//      }
    }

    @Override
    public void init() {
        mTeamId = getIntent().getStringExtra(TEAM_ID);
        if (TextUtils.isEmpty(mTeamId)) {
            interrupt();
            return;
        }

        mTeam = NimTeamSDK.queryTeamBlock(mTeamId);
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_team_name_set);
        ButterKnife.inject(this);

        initToolbar();
        mEtName.setText(mTeam.getName());
        mEtName.setSelection(mTeam.getName().length());
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }
```

```
    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("");
        mToolbar.setNavigationIcon(R.mipmap.ic_back);

        mBtnOk.setVisibility(View.VISIBLE);
        mBtnOk.setText("");
    }
}
```

46:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\TestActivity.java

```
package com.lqr.wechat.activity;


import com.lqr.wechat.R;

/**
 * @ CSDN_LQR
 * @
 */
public class TestActivity extends BaseActivity {


    @Override
    public void initView() {
        setContentView(R.layout.activity_test);
    }

}
```

47:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\TransferActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.MenuItem;
import android.widget.EditText;
```

```
import com.lqr.wechat.R;
import com.lqr.wechat.utils.UIUtils;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class TransferActivity extends BaseActivity {

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @InjectView(R.id.etMoney)
    EditText mEtMoney;

    @Override
    public void initView() {
        setContentView(R.layout.activity_transfer);
        ButterKnife.inject(this);
        initToolbar();
    }

    @Override
    public void initListener() {
        mEtMoney.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                if (s.length() > 12) {
                    mEtMoney.setText(s.subSequence(0, 12));
                    mEtMoney.setSelection(12);
                }
            }

            @Override
```

```
            public void afterTextChanged(Editable s) {

            }
        });
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setSubtitle("");
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }

}
```

48:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\UserInfoActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
```

```
import android.widget.RelativeLayout;
import android.widget.ScrollView;
import android.widget.TextView;

import com.lqr.optionitemview.OptionItemView;
import com.lqr.wechat.AppConst;
import com.lqr.wechat.R;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.Contact;
import com.lqr.wechat.model.UserCache;
import com.lqr.wechat.nimsdk.NimBlackListSDK;
import com.lqr.wechat.nimsdk.NimFriendSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.StringUtils;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.uinfo.constant.GenderEnum;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.List;
import java.util.Map;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class UserInfoActivity extends BaseActivity {

    public static final String USER_INFO_ACCOUNT = "account";

    private Intent mIntent;
    private Animation mPushBottomInAnimation;
    private Animation mPushBottomOutAnimation;
    private String mAccount;
    private Contact mContact;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
```

```
    //
    @InjectView(R.id.ivHeader)
    ImageView mIvHeader;
    @InjectView(R.id.tvAlias)
    TextView mTvAlias;
    @InjectView(R.id.tvAccount)
    TextView mTvAccount;
    @InjectView(R.id.tvName)
    TextView mTvName;
    @InjectView(R.id.ivGender)
    ImageView mIvGender;

    @InjectView(R.id.oivAliasAndTag)
    OptionItemView mOivAliasAndTag;
    @InjectView(R.id.llArea)
    LinearLayout mLlArea;
    @InjectView(R.id.tvArea)
    TextView mTvArea;
    @InjectView(R.id.llSignature)
    LinearLayout mLlSignature;
    @InjectView(R.id.tvSignature)
    TextView mTvSignature;

    @InjectView(R.id.btnCheat)
    Button mBtnCheat;
    @InjectView(R.id.btnVideoCheat)
    Button mBtnVideoCheat;
    @InjectView(R.id.btnAddFriend)
    Button mBtnAddFriend;

    //
    @InjectView(R.id.rlMenu)
    RelativeLayout mRlMenu;
    @InjectView(R.id.vMask)
    View mVMask;
    @InjectView(R.id.svMenu)
    ScrollView mSvMenu;

    @OnClick({R.id.oivAliasAndTag, R.id.btnCheat, R.id.btnVideoCheat, R.id.btnAddFriend,
R.id.oivAlias, R.id.oivFriendsCirclePrivacySet, R.id.oivAddToBlackList, R.id.oivDelete})
    public void click(View view) {
```

```
        switch (view.getId()) {
            case R.id.oivAliasAndTag:
                jumpToAliasActivity();
                break;
            case R.id.btnCheat:
                setResult(RESULT_OK);
                mIntent = new Intent(this, SessionActivity.class);
                mIntent.putExtra(SessionActivity.SESSION_ACCOUNT, mAccount);
                startActivity(mIntent);
                finish();
                break;
            case R.id.btnVideoCheat:
                break;
            case R.id.btnAddFriend:
                mIntent = new Intent(this, PostscriptActivity.class);
                mIntent.putExtra("account", mAccount);
                startActivity(mIntent);
                break;
            case R.id.oivAlias://
                jumpToAliasActivity();
                hideMenu();
                break;
            case R.id.oivFriendsCirclePrivacySet://
                startActivity(new Intent(UserInfoActivity.this, FriendCirclePrivacySetActivity.class));
                hideMenu();
                break;
            case R.id.oivAddToBlackList://
                hideMenu();
                showMaterialDialog("", "", "", "", new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        NimBlackListSDK.addToBlackList(mAccount, new RequestCallback<Void>() {
                            @Override
                            public void onSuccess(Void param) {
                                UIUtils.showToast("");
                                Intent intent = new Intent(UserInfoActivity.this, MainActivity.class);
                                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
                                startActivity(intent);
                            }

                            @Override
```

```
                public void onFailed(int code) {
                    UIUtils.showToast("" + code);
                }

                @Override
                public void onException(Throwable exception) {
                    exception.printStackTrace();
                }
            });
            hideMaterialDialog();
        }
    }, new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            hideMaterialDialog();
        }
    });
    break;
case R.id.oivDelete://
    hideMenu();
    showMaterialDialog("", "" + mContact.getDisplayName() + "", "", "", new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //
            NimFriendSDK.deleteFriend(mAccount, new RequestCallback<Void>() {
                @Override
                public void onSuccess(Void param) {
                    UIUtils.showToast("");
                    Intent intent = new Intent(UserInfoActivity.this, MainActivity.class);
                    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
                    startActivity(intent);
                }

                @Override
                public void onFailed(int code) {
                    UIUtils.showToast("" + code);
                }

                @Override
                public void onException(Throwable exception) {
```

```
                    exception.printStackTrace();
                }
            });
            hideMaterialDialog();
        }
    }, new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //
            hideMaterialDialog();
        }
    });
    break;
  }
}

private void jumpToAliasActivity() {
  mIntent = new Intent(UserInfoActivity.this, AliasActivity.class);
  mIntent.putExtra("contact", mContact);
  startActivityForResult(mIntent, AliasActivity.REQ_CHANGE_ALIAS);
}

@OnClick(R.id.vMask)
public void mask() {
  toggleMenu();
}

@Override
public void init() {
  mAccount = getIntent().getStringExtra("account");
  if (TextUtils.isEmpty(mAccount)) {
    interrupt();
    return;
  }
}

@Override
public void initView() {
  setContentView(R.layout.activity_user_info);
  ButterKnife.inject(this);
  initToolbar();
  initAnimation();
```

```
        if (UserCache.getAccount().equals(mAccount)) {//
            mOivAliasAndTag.setVisibility(View.GONE);
            mLlArea.setVisibility(View.GONE);
            mLlSignature.setVisibility(View.GONE);
            mBtnCheat.setVisibility(View.VISIBLE);
        } else {
            if (NimFriendSDK.isMyFriend(mAccount)) {
                mBtnCheat.setVisibility(View.VISIBLE);
                mOivAliasAndTag.setVisibility(View.VISIBLE);
//                mBtnVideoCheat.setVisibility(View.VISIBLE);
                mBtnAddFriend.setVisibility(View.GONE);
            } else {
                mBtnCheat.setVisibility(View.GONE);
                mOivAliasAndTag.setVisibility(View.GONE);
//                mBtnVideoCheat.setVisibility(View.GONE);
                mBtnAddFriend.setVisibility(View.VISIBLE);
            }
        }
    }

    @Override
    public void initData() {
        //
        if (NimFriendSDK.isMyFriend(mAccount)) {
            mContact = new Contact(mAccount);
            //
            setUserInfo();
            //
            getUserInfoFromServer();
        } else {
            getUserInfoFromServer();
        }
    }

    private void getUserInfoFromServer() {
        NimUserInfoSDK.getUserInfoFromServer(mAccount, new
RequestCallback<List<NimUserInfo>>() {
            @Override
            public void onSuccess(List<NimUserInfo> param) {
                if (param != null && param.size() > 0) {
                    mContact = new Contact(NimFriendSDK.getFriendByAccount(mAccount),
```

```
param.get(0));
                //
                setUserInfo();
            }
        }

        @Override
        public void onFailed(int code) {

        }

        @Override
        public void onException(Throwable exception) {

        }
    });
}


@Override
public boolean onCreateOptionsMenu(Menu menu) {
    if (NimFriendSDK.isMyFriend(mAccount)) {
        new MenuInflater(this).inflate(R.menu.menu_more, menu);
    }
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            onBackPressed();
            break;
        case R.id.itemMore:
            toggleMenu();
            break;
    }
    return super.onOptionsItemSelected(item);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
    if (requestCode == AliasActivity.REQ_CHANGE_ALIAS && resultCode == RESULT_OK) {
        //
        initData();
    }
    super.onActivityResult(requestCode, resultCode, data);
}

@Override
public void onBackPressed() {
    if (mRlMenu.getVisibility() == View.VISIBLE) {
        //
        mSvMenu.startAnimation(mPushBottomOutAnimation);
        return;
    }
    super.onBackPressed();
}

private void initToolbar() {
    setSupportActionBar(mToolbar);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setTitle("");
    mToolbar.setNavigationIcon(R.mipmap.ic_back);
}

private void initAnimation() {
    mPushBottomInAnimation = AnimationUtils.loadAnimation(this, R.anim.push_bottom_in);
    mPushBottomOutAnimation = AnimationUtils.loadAnimation(this, R.anim.push_bottom_out);
    mPushBottomInAnimation.setDuration(300);
    mPushBottomOutAnimation.setDuration(300);
    mPushBottomOutAnimation.setAnimationListener(new Animation.AnimationListener() {
        @Override
        public void onAnimationStart(Animation animation) {

        }

        @Override
        public void onAnimationEnd(Animation animation) {
            mRlMenu.setVisibility(View.GONE);
        }

        @Override
        public void onAnimationRepeat(Animation animation) {
```

```
            }
        });
    }

    private void toggleMenu() {
        if (mRlMenu.getVisibility() == View.VISIBLE) {
            //
            mSvMenu.startAnimation(mPushBottomOutAnimation);
        } else {
            //
            mRlMenu.setVisibility(View.VISIBLE);
            mSvMenu.startAnimation(mPushBottomInAnimation);
        }
    }

    private void hideMenu() {
        mSvMenu.startAnimation(mPushBottomOutAnimation);
    }

    private void setUserInfo() {
        //
        if (TextUtils.isEmpty(mContact.getAvatar())) {
            mIvHeader.setImageResource(R.mipmap.default_header);
        } else {
            ImageLoaderManager.LoadNetImage(mContact.getAvatar(), mIvHeader);
        }

        //
        NimUserInfo userInfo = mContact.getUserInfo();
        if (userInfo.getGenderEnum() == GenderEnum.FEMALE) {
            mIvGender.setImageResource(R.mipmap.ic_gender_female);
        } else if (userInfo.getGenderEnum() == GenderEnum.MALE) {
            mIvGender.setImageResource(R.mipmap.ic_gender_male);
        } else {
            mIvGender.setVisibility(View.GONE);
        }

        //
        if (TextUtils.isEmpty(mContact.getAlias())) {
//            mTvAlias.setVisibility(View.GONE);
            mTvName.setVisibility(View.GONE);
```

```
        } else {
//            mTvAlias.setVisibility(View.VISIBLE);
            mTvName.setVisibility(View.VISIBLE);
        }
        mTvAlias.setText(mContact.getDisplayName());
        mTvAccount.setText(":" + mContact.getAccount());
        mTvName.setText(":" + mContact.getName());
        Map<String, Object> extensionMap = mContact.getUserInfo().getExtensionMap();
        if (extensionMap != null)
            mTvArea.setText(StringUtils.isEmpty(extensionMap.get(AppConst.UserInfoExt.AREA)) ?
"" : (String) extensionMap.get(AppConst.UserInfoExt.AREA));
        mTvSignature.setText(mContact.getUserInfo().getSignature());
    }
}
```

49:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\VipCardActivity.java

```
package com.lqr.wechat.activity;

import android.support.v7.widget.Toolbar;
import android.view.MenuItem;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @ --
 */
public class VipCardActivity extends BaseActivity {
    @InjectView(R.id.toolbar)
    Toolbar mToolbar;

    @Override
    public void initView() {
        setContentView(R.layout.activity_vip_card);
        ButterKnife.inject(this);
        initToolbar();
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void initToolbar() {
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        mToolbar.setNavigationIcon(R.mipmap.ic_back);
    }
}
```

50:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\activity\WebViewActivity.java

```
package com.lqr.wechat.activity;

import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.MenuItem;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

import com.lqr.wechat.R;
import com.lqr.wechat.utils.StringUtils;
import com.lqr.wechat.view.ProgressWebView;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
```

```
 */
public class WebViewActivity extends BaseActivity {

    private Intent mIntent;
    private Bundle mExtras;
    private String mUrl;
    private String mTitle;

    private boolean isLoading = false;

    @InjectView(R.id.toolbar)
    Toolbar mToolbar;
    @InjectView(R.id.webview)
    public ProgressWebView mWebView;

    @Override
    public void init() {
        //url
        try {
            mIntent = getIntent();
            mExtras = mIntent.getExtras();
            if (mExtras == null) {
                finish();
                return;
            }
            mUrl = mExtras.getString("url");
            if (StringUtils.isEmpty(mUrl)) {
                finish();
                return;
            }
            mTitle = mExtras.getString("title");
        } catch (Exception e) {
            e.printStackTrace();
            finish();
            return;
        }
    }

    @Override
    public void initView() {
        setContentView(R.layout.activity_webview);
        ButterKnife.inject(this);
```

```
    initToolbar();

    //webView
    WebSettings settings = mWebView.getSettings();
    settings.setRenderPriority(WebSettings.RenderPriority.HIGH);
    settings.setSupportMultipleWindows(true);
    settings.setJavaScriptEnabled(true);
    settings.setSavePassword(false);
    settings.setJavaScriptCanOpenWindowsAutomatically(true);
    settings.setMinimumFontSize(settings.getMinimumLogicalFontSize() + 8);
    settings.setAllowFileAccess(false);
    settings.setTextSize(WebSettings.TextSize.NORMAL);
    mWebView.setVerticalScrollbarOverlay(true);
    mWebView.setWebViewClient(new MyWebViewClient());
    mWebView.loadUrl(mUrl);
}

@Override
public void initListener() {
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            break;
    }
    return super.onOptionsItemSelected(item);
}

@Override
public void onBackPressed() {
    isLoading = false;

    //
    if (mWebView.canGoBack()) {
        mWebView.goBack();
    } else {
        finish();
    }
}
```

```
@Override
protected void onDestroy() {
    super.onDestroy();
    if (mWebView != null) {
        mWebView.removeAllViews();
        try {
            mWebView.destroy();
        } catch (Exception e) {
            e.printStackTrace();
        }
        mWebView = null;
    }
}

private void initToolbar() {
    setSupportActionBar(mToolbar);
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setTitle(TextUtils.isEmpty(mTitle) ? "" : mTitle);
    mToolbar.setNavigationIcon(R.mipmap.ic_delete_white);
}

private class MyWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        //
        mWebView.loadUrl(url);
        return true;
    }

    @Override
    public void onPageStarted(WebView view, String url, Bitmap favicon) {
        super.onPageStarted(view, url, favicon);
        isLoading = true;
    }

    @Override
    public void onPageFinished(WebView view, String url) {
        super.onPageFinished(view, url);
        isLoading = false;
    }
}
```

```
}
```

51:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\adapter\FuncPagerAdapter.java

```
package com.lqr.wechat.adapter;

import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;

import com.lqr.wechat.fragment.BaseFragment;

import java.util.List;

/**
 * @ CSDN_LQR
 * @
 */
public class FuncPagerAdapter extends FragmentPagerAdapter {

    private List<BaseFragment> mFragments;

    public FuncPagerAdapter(FragmentManager fm, List<BaseFragment> fragments) {
        super(fm);
        mFragments = fragments;
    }

    @Override
    public Fragment getItem(int position) {
        return mFragments.get(position);
    }

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

52:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\adapter\MainPagerAdapter.java

```
package com.lqr.wechat.adapter;
```

```
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;

import com.lqr.wechat.fragment.BaseFragment;

import java.util.List;

/**
 * @ CSDN_LQR
 * @ ViewPager
 */
public class MainPagerAdapter extends FragmentPagerAdapter {

    private List<BaseFragment> mFragments;

    public MainPagerAdapter(FragmentManager fm, List<BaseFragment> fragments) {
        super(fm);
        mFragments = fragments;
    }

    @Override
    public Fragment getItem(int position) {
        return mFragments.get(position);
    }

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

53:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\adapter\SessionAdapter.java

```
package com.lqr.wechat.adapter;

import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.drawable.AnimationDrawable;
import android.text.TextUtils;
```

```
import android.text.style.ImageSpan;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.RelativeLayout;

import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
import com.lqr.emoji.MoonUtil;
import com.lqr.emoji.StickerManager;
import com.lqr.wechat.R;
import com.lqr.wechat.activity.FilePreviewActivity;
import com.lqr.wechat.activity.ImageWatchActivity;
import com.lqr.wechat.activity.SessionActivity;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.nimsdk.NimMessageSDK;
import com.lqr.wechat.nimsdk.NimTeamSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.nimsdk.audio.BaseAudioControl;
import com.lqr.wechat.nimsdk.audio.MessageAudioControl;
import com.lqr.wechat.nimsdk.audio.Playable;
import com.lqr.wechat.nimsdk.custom.StickerAttachment;
import com.lqr.wechat.nimsdk.utils.ImageUtil;
import com.lqr.wechat.nimsdk.utils.ScreenUtil;
import com.lqr.wechat.utils.Bimp;
import com.lqr.wechat.utils.FileIconUtils;
import com.lqr.wechat.utils.FileOpenUtils;
import com.lqr.wechat.utils.FileUtils;
import com.lqr.wechat.utils.LogUtils;
import com.lqr.wechat.utils.MimeTypeUtils;
import com.lqr.wechat.utils.TimeUtils;
import com.lqr.wechat.utils.UIUtils;
import com.lqr.wechat.utils.VideoThumbLoader;
import com.lqr.wechat.view.BubbleImageView;
import com.lqr.wechat.view.CircularProgressBar;
import com.netease.nimlib.sdk.AbortableFuture;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.media.record.AudioRecorder;
import com.netease.nimlib.sdk.msg.attachment.AudioAttachment;
import com.netease.nimlib.sdk.msg.attachment.FileAttachment;
import com.netease.nimlib.sdk.msg.attachment.ImageAttachment;
import com.netease.nimlib.sdk.msg.attachment.NotificationAttachment;
```

```
import com.netease.nimlib.sdk.msg.attachment.VideoAttachment;
import com.netease.nimlib.sdk.msg.constant.AttachStatusEnum;
import com.netease.nimlib.sdk.msg.constant.MsgDirectionEnum;
import com.netease.nimlib.sdk.msg.constant.MsgStatusEnum;
import com.netease.nimlib.sdk.msg.constant.MsgTypeEnum;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.msg.model.IMMessage;
import com.netease.nimlib.sdk.team.model.MemberChangeAttachment;
import com.netease.nimlib.sdk.team.model.MuteMemberAttachment;
import com.netease.nimlib.sdk.team.model.UpdateTeamAttachment;
import com.zhy.http.okhttp.OkHttpUtils;
import com.zhy.http.okhttp.callback.FileCallBack;

import java.io.File;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import okhttp3.Call;

import static com.netease.nimlib.sdk.msg.constant.MsgTypeEnum.notification;

/**
 * @ CSDN_LQR
 * @
 */
public class SessionAdapter extends LQRAdapterForRecyclerView<IMMessage> {

    public static final int CLICK_TO_PLAY_AUDIO_DELAY = 500;
    private Context mContext;

    private static final int NOTIFICATION = R.layout.item_notification;
    private static final int SEND_TEXT = R.layout.item_text_send;
    private static final int RECEIVE_TEXT = R.layout.item_text_receive;
    private static final int SEND_STICKER = R.layout.item_sticker_send;
    private static final int RECEIVE_STICKER = R.layout.item_sticker_receive;
    private static final int SEND_IMAGE = R.layout.item_image_send;
    private static final int RECEIVE_IMAGE = R.layout.item_image_receive;
    private static final int SEND_VIDEO = R.layout.item_video_send;
    private static final int RECEIVE_VIDEO = R.layout.item_video_receive;
    private static final int SEND_LOCATION = R.layout.item_location_send;
    private static final int RECEIVE_LOCATION = R.layout.item_location_receive;
```

```
    private static final int SEND_AUDIO = R.layout.item_audio_send;
    private static final int RECEIVE_AUDIO = R.layout.item_audio_receive;
    private static final int SEND_FILE = R.layout.item_file_send;
    private static final int RECEIVE_FILE = R.layout.item_file_receive;

    private Map<String, Float> mProgress = new HashMap<>();
    private MessageAudioControl mAudioControl;
    private ImageView mAnimationView;

    public SessionAdapter(Context context, List<IMMessage> data) {
        super(context, data);
        mContext = context;
        mAudioControl = MessageAudioControl.getInstance(mContext);
    }

    public SessionAdapter(Context context, int defaultLayoutId, List<IMMessage> data) {
        super(context, defaultLayoutId, data);
    }

    @Override
    public void convert(LQRViewHolderForRecyclerView helper, final IMMessage item, final int
position) {

        //
        setTime(helper, item, position);

        if (item.getMsgType() != notification) {
            //
            setHeader(helper, item);

            //
            if (item.getSessionType() == SessionTypeEnum.Team) {
                helper.setViewVisibility(R.id.tvName,
NimTeamSDK.shouldShowNickName(item.getSessionId()) ? View.VISIBLE : View.GONE)
                        .setText(R.id.tvName,
NimTeamSDK.getTeamMemberDisplayNameWithoutMe(item.getSessionId(),
item.getFromAccount()));
            } else {
                helper.setViewVisibility(R.id.tvName, View.GONE);
            }

            //
```

```
        helper.getView(R.id.ivError).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                NimMessageSDK.reSendMessage(item);
//              notifyItemChanged(position);
//              notifyDataSetChanged();
                ((SessionActivity) mContext).initData();
            }
        });

        //
        setViewWithStatus(helper, item, position);
    }

    //
    if (item.getMsgType() == MsgTypeEnum.text) {
        setTextMessage(helper, item);
    }
    //
    else if (item.getMsgType() == MsgTypeEnum.custom) {
        setStickerMessage(helper, item);
    }
    //
    else if (item.getMsgType() == MsgTypeEnum.image) {
        setImageMessage(helper, item);
    }
    //
    else if (item.getMsgType() == MsgTypeEnum.audio) {
        setAudioMessage(helper, item);
    }
    //
    else if (item.getMsgType() == MsgTypeEnum.video) {
        setVideoMessage(helper, item, position);
    }
    //
    else if (item.getMsgType() == MsgTypeEnum.file) {
        setFileMessage(helper, item);
    }
    //
    else if (item.getMsgType() == notification) {
        setNotificationMessage(helper, item);
    }
```

```
    }

    private void setTime(LQRViewHolderForRecyclerView helper, IMMessage item, int position) {
        if (position > 0) {
            IMMessage preMessage = getData().get(position - 1);
            if (item.getTime() - preMessage.getTime() > (5 * 60 * 1000)) {//5
                helper.setViewVisibility(R.id.tvTime, View.VISIBLE).setText(R.id.tvTime,
TimeUtils.getMsgFormatTime(item.getTime()));
            } else {
                helper.setViewVisibility(R.id.tvTime, View.GONE);
            }
        } else {
            helper.setViewVisibility(R.id.tvTime, View.VISIBLE).setText(R.id.tvTime,
TimeUtils.getMsgFormatTime(item.getTime()));
        }
    }

    private void setHeader(LQRViewHolderForRecyclerView helper, IMMessage item) {
        ImageView ivAvatar = helper.getView(R.id.ivAvatar);
        String avatar = NimUserInfoSDK.getUser(item.getFromAccount()).getAvatar();
        if (!TextUtils.isEmpty(avatar)) {
            ImageLoaderManager.LoadNetImage(avatar, ivAvatar);
        } else {
            ivAvatar.setImageResource(R.mipmap.default_header);
        }
    }

    private void setTextMessage(LQRViewHolderForRecyclerView helper, IMMessage item) {
        helper.setText(R.id.tvText, item.getContent());
        //
        MoonUtil.identifyFaceExpression(UIUtils.getContext(), helper.getView(R.id.tvText),
item.getContent(), ImageSpan.ALIGN_BOTTOM);
    }

    private void setStickerMessage(LQRViewHolderForRecyclerView helper, IMMessage item) {
        StickerAttachment attachment = (StickerAttachment) item.getAttachment();
        String uri = StickerManager.getInstance().getStickerBitmapUri(attachment.getCatalog(),
attachment.getChartlet());
        ImageLoaderManager.LoadNetImage(uri, (ImageView) helper.getView(R.id.ivSticker));
    }
```

```
    private void setImageMessage(LQRViewHolderForRecyclerView helper, final IMMessage item)
{
        final BubbleImageView bivPic = helper.getView(R.id.bivPic);
        final ImageAttachment ia = (ImageAttachment) item.getAttachment();

        if (!TextUtils.isEmpty(ia.getPath())) {
            ImageLoaderManager.LoadLocalImage(ia.getPath(), bivPic);
        } else {
            //
            if (ia.getThumbPath() == null) {
                LogUtils.sf("");
                AbortableFuture abortableFuture = NimMessageSDK.downloadAttachment(item, true);
                abortableFuture.setCallback(new RequestCallback() {
                    @Override
                    public void onSuccess(Object param) {
                        Bitmap bitmap = Bimp.getLoacalBitmap(ia.getThumbPath());
                        if (bitmap != null) {
                            bivPic.setImageBitmap(bitmap);
                        }
                    }

                    @Override
                    public void onFailed(int code) {

                    }

                    @Override
                    public void onException(Throwable exception) {

                    }
                });
            } else {
                LogUtils.sf("");
                Bitmap bitmap = Bimp.getLoacalBitmap(ia.getThumbPath());
                if (bitmap != null) {
                    bivPic.setImageBitmap(bitmap);
                }
            }
        }

        //
        bivPic.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(mContext, ImageWatchActivity.class);
                intent.putExtra("message", item);
                intent.putExtra("account", ((SessionActivity) mContext).mSessionId);
                intent.putExtra("sessionType", ((SessionActivity) mContext).mSessionType);

                mContext.startActivity(intent);
            }
        });
    }

    /**
     *
     */
    private void setAudioMessage(final LQRViewHolderForRecyclerView helper, final IMMessage
item) {
        final AudioAttachment aa = (AudioAttachment) item.getAttachment();
        long durationMillis = aa.getDuration();
        long durationSecond = durationMillis / 1000;
        int increment = (int) (ScreenUtil.getDisplayWidth() / 2 /
AudioRecorder.DEFAULT_MAX_AUDIO_RECORD_TIME_SECOND * durationSecond);

        //
        RelativeLayout rlAudio = helper.setText(R.id.tvDuration, durationSecond +
"'").getView(R.id.rlAudio);
        ViewGroup.LayoutParams params = rlAudio.getLayoutParams();
        params.width = UIUtils.dip2Px(65) + UIUtils.dip2Px(increment);
        rlAudio.setLayoutParams(params);

        //
        helper.getView(R.id.rlAudio).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //
                mAudioControl.stopAudio();
                //
                mAnimationView = helper.getView(R.id.ivAudio);
                //
                if (TextUtils.isEmpty(aa.getPath())) {
                    OkHttpUtils.get().url(aa.getUrl()).build().execute(new
FileCallBack(FileUtils.getDirFromPath(aa.getPathForSave()),
```

```
FileUtils.getFileNameFromPath(aa.getPathForSave())) {
                    @Override
                    public void onError(Call call, Exception e, int id) {
                        UIUtils.showToast("");
                    }

                    @Override
                    public void onResponse(File response, int id) {
                        playAudioDelayAndSetPlayNext(item);
                    }
                });
            } else {
                playAudioDelayAndSetPlayNext(item);
            }

        }
    });
}

/**
 *
 */
private void setVideoMessage(LQRViewHolderForRecyclerView helper, final IMMessage item,
final int position) {
    final BubbleImageView bivPic = helper.getView(R.id.bivPic);
    final VideoAttachment va = (VideoAttachment) item.getAttachment();

    //
    int[] bounds = new int[]{va.getWidth(), va.getHeight()};
    final ImageUtil.ImageSize imageSize = ImageUtil.getThumbnailDisplaySize(bounds[0],
bounds[1], getImageMaxEdge(), getImageMinEdge());
    setLayoutParams(imageSize.width, imageSize.height, bivPic);

    //
    if (!TextUtils.isEmpty(va.getPath())) {
        VideoThumbLoader.getInstance().showThumb(va.getPath(), bivPic, imageSize.width,
imageSize.height);
    } else {
        bivPic.setImageResource(R.mipmap.img_video_default);
        //
        AbortableFuture abortableFuture = NimMessageSDK.downloadAttachment(item, true);
        abortableFuture.setCallback(new RequestCallback() {
```

```
            @Override
            public void onSuccess(Object param) {
                Bitmap bitmap = Bimp.getLoacalBitmap(va.getThumbPath());
                if (bitmap != null) {
                    bivPic.setImageBitmap(bitmap);
                }
            }

            @Override
            public void onFailed(int code) {

            }

            @Override
            public void onException(Throwable exception) {

            }
        });
    }

    //
    helper.getView(R.id.ivPlay).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //
            if (!TextUtils.isEmpty(va.getPath())) {
                //
                FileOpenUtils.openFile(mContext, va.getPath(),
MimeTypeUtils.getMimeType(va.getDisplayName()));
            } else {
                //
                OkHttpUtils.get().url(va.getUrl()).build().execute(new
FileCallBack(FileUtils.getDirFromPath(va.getPathForSave()),
FileUtils.getFileNameFromPath(va.getPathForSave())) {
                    @Override
                    public void onError(Call call, Exception e, int id) {
                        UIUtils.showToast("");
                    }

                    @Override
                    public void onResponse(File response, int id) {
                        if (!TextUtils.isEmpty(va.getPath())) {
```

```
                        VideoThumbLoader.getInstance().showThumb(va.getPath(), bivPic,
imageSize.width, imageSize.height);
                        notifyItemChanged(position);
                    }
                }

                @Override
                public void inProgress(float progress, long total, int id) {//use progress: 0 ~ 1
                    progress = progress * 100;
                    putProgress(item, progress);
                    if (progress == 0) {
                        item.setAttachStatus(AttachStatusEnum.def);
                    } else if (progress < 100) {
                        item.setAttachStatus(AttachStatusEnum.transferring);
                    } else {
                        item.setAttachStatus(AttachStatusEnum.transferred);
                    }
                    notifyItemChanged(position);
                    super.inProgress(progress, total, id);
                }
            });
        }
    }
});
}

/**
 *
 */
private void setFileMessage(LQRViewHolderForRecyclerView helper, final IMMessage item) {
    FileAttachment fa = (FileAttachment) item.getAttachment();
    helper.setImageResource(R.id.ivIcon, FileIconUtils.getFileIconResId(fa.getExtension()))
            .setText(R.id.tvFileName, fa.getFileName())
            .setText(R.id.tvFileSize, FileUtils.formateFileSize(fa.getSize()))
            .getView(R.id.llFile).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //
            Intent intent = new Intent(mContext, FilePreviewActivity.class);
            intent.putExtra("message", item);
            mContext.startActivity(intent);
        }
```

```
        });
    }

    /**
     *
     */
    private void setNotificationMessage(LQRViewHolderForRecyclerView helper, IMMessage item)
{
        NotificationAttachment na = (NotificationAttachment) item.getAttachment();
        String fromAccount = item.getFromAccount();
        String text = "";
        switch (na.getType()) {
            case InviteMember:
                text = NimTeamSDK.buildInviteMemberNotification(((MemberChangeAttachment) na),
item.getSessionId(), fromAccount);
                break;
            case KickMember:
                text = NimTeamSDK.buildKickMemberNotification(((MemberChangeAttachment) na),
item.getSessionId(), fromAccount);
                break;
            case LeaveTeam:
                text = NimTeamSDK.buildLeaveTeamNotification(item.getSessionId(), fromAccount);
                break;
            case DismissTeam:
                text = NimTeamSDK.buildDismissTeamNotification(item.getSessionId(), fromAccount);
                break;
            case UpdateTeam:
                text = NimTeamSDK.buildUpdateTeamNotification(((UpdateTeamAttachment) na),
item.getSessionId(), fromAccount);
                break;
            case PassTeamApply:
                text =
NimTeamSDK.buildManagerPassTeamApplyNotification((MemberChangeAttachment) na,
item.getSessionId());
                break;
            case TransferOwner:
                text = NimTeamSDK.buildTransferOwnerNotification(((MemberChangeAttachment) na),
item.getSessionId(), fromAccount);
                break;
            case AddTeamManager:
                text = NimTeamSDK.buildAddTeamManagerNotification((MemberChangeAttachment)
na, item.getSessionId());
```

```
                break;
            case RemoveTeamManager:
                text =
NimTeamSDK.buildRemoveTeamManagerNotification((MemberChangeAttachment) na,
item.getSessionId());
                break;
            case AcceptInvite:
                text = NimTeamSDK.buildAcceptInviteNotification(((MemberChangeAttachment) na),
item.getSessionId(), fromAccount);
                break;
            case MuteTeamMember:
                text = NimTeamSDK.buildMuteTeamNotification((MuteMemberAttachment) na,
item.getSessionId());
                break;
            default:
                text = NimTeamSDK.getTeamMemberDisplayNameWithYou(item.getSessionId(),
fromAccount) + ": unknown message";
                break;
        }
        helper.setText(R.id.tvNotification, text);
    }

    private void setViewWithStatus(LQRViewHolderForRecyclerView helper, IMMessage item, final
int position) {
        ///
        MsgStatusEnum status = item.getStatus();

        if (status == MsgStatusEnum.success) {
            LogUtils.sf("...");
            helper.setViewVisibility(R.id.llError, View.GONE);
            setProgressVisiable(helper, item, false);
        } else if (status == MsgStatusEnum.fail) {
            LogUtils.sf("...");
            helper.setViewVisibility(R.id.llError, View.VISIBLE);
            setProgressVisiable(helper, item, false);
        } else if (status == MsgStatusEnum.sending) {
            LogUtils.sf("...");
            helper.setViewVisibility(R.id.llError, View.GONE);
            setProgressVisiable(helper, item, true);
            //
            updateProgress(helper, item);
```

```
        //success
        UIUtils.postTaskDelay(new Runnable() {
            @Override
            public void run() {
                notifyItemChanged(position);
            }
        }, 1000);
    }

    ///
    if (item.getAttachment() != null) {
        AttachStatusEnum attachStatus = item.getAttachStatus();
        if (attachStatus == AttachStatusEnum.def) {

        } else if (attachStatus == AttachStatusEnum.transferring) {
            setProgressVisiable(helper, item, true);
            if (item.getMsgType() == MsgTypeEnum.video)
                helper.setViewVisibility(R.id.ivPlay, View.GONE);
            //
            updateProgress(helper, item);
        } else if (attachStatus == AttachStatusEnum.transferred) {
            setProgressVisiable(helper, item, false);
            if (item.getMsgType() == MsgTypeEnum.video)
                helper.setViewVisibility(R.id.ivPlay, View.VISIBLE);
        } else if (attachStatus == AttachStatusEnum.fail) {
            setProgressVisiable(helper, item, false);
            if (item.getMsgType() == MsgTypeEnum.video)
                helper.setViewVisibility(R.id.ivPlay, View.GONE);
        }
    }
}

private void updateProgress(LQRViewHolderForRecyclerView helper, IMMessage item) {
    if (item.getAttachment() instanceof ImageAttachment) {
        //
        BubbleImageView bivPic = helper.getView(R.id.bivPic);
        Float progress = getProgress(item);
        if (progress != null) {
            bivPic.setPercent((int) progress.floatValue());
            LogUtils.sf("" + getProgress(item) + "%");
        }
    } else if (item.getAttachment() instanceof VideoAttachment) {
```

```
            //
            CircularProgressBar cpbLoading = helper.getView(R.id.cpbLoading);
            Float progress = getProgress(item);
            if (progress != null) {
                cpbLoading.setProgress(progress.intValue());
                LogUtils.sf("" + getProgress(item) + "%");
            }
        }
    }

    private void setProgressVisiable(LQRViewHolderForRecyclerView helper, IMMessage item,
boolean visiable) {
        if (visiable) {
            if (item.getMsgType() == MsgTypeEnum.text || item.getMsgType() ==
MsgTypeEnum.custom || item.getMsgType() == MsgTypeEnum.location) {
                helper.setViewVisibility(R.id.pbSending, View.VISIBLE);
            } else if (item.getMsgType() == MsgTypeEnum.image || item.getMsgType() ==
MsgTypeEnum.video) {
                BubbleImageView bivPic = helper.getView(R.id.bivPic);
                bivPic.showShadow(true);
                if (item.getMsgType() == MsgTypeEnum.image)
                    bivPic.setProgressVisible(true);
                else
                    helper.setViewVisibility(R.id.cpbLoading, View.VISIBLE).setViewVisibility(R.id.ivPlay,
View.GONE);
            }
        } else {
            if (item.getMsgType() == MsgTypeEnum.text || item.getMsgType() ==
MsgTypeEnum.custom || item.getMsgType() == MsgTypeEnum.location) {
                helper.setViewVisibility(R.id.pbSending, View.GONE);
            } else if (item.getMsgType() == MsgTypeEnum.image || item.getMsgType() ==
MsgTypeEnum.video) {
                BubbleImageView bivPic = helper.getView(R.id.bivPic);
                bivPic.showShadow(false);
                if (item.getMsgType() == MsgTypeEnum.image)
                    bivPic.setProgressVisible(false);
                else
                    helper.setViewVisibility(R.id.cpbLoading, View.GONE).setViewVisibility(R.id.ivPlay,
View.VISIBLE);
            }
        }
    }
```

```
/**
 *
 */
public void putProgress(IMMessage message, float progress) {
    mProgress.put(message.getUuid(), progress);
}

/**
 *
 */
public Float getProgress(IMMessage message) {
    return mProgress.get(message.getUuid());
}

@Override
public int getItemViewType(int position) {
    IMMessage msg = getData().get(position);
    MsgTypeEnum msgType = msg.getMsgType();
    if (msgType == notification) {
        return NOTIFICATION;
    }
    if (msgType == MsgTypeEnum.text) {
        if (msg.getDirect() == MsgDirectionEnum.Out) {
            return SEND_TEXT;
        } else {
            return RECEIVE_TEXT;
        }
    }
    if (msgType == MsgTypeEnum.custom) {
        if (msg.getDirect() == MsgDirectionEnum.Out) {
            return SEND_STICKER;
        } else {
            return RECEIVE_STICKER;
        }
    }
    if (msgType == MsgTypeEnum.image) {
        if (msg.getDirect() == MsgDirectionEnum.Out) {
            return SEND_IMAGE;
        } else {
            return RECEIVE_IMAGE;
        }
```

```
        }
        if (msgType == MsgTypeEnum.video) {
            if (msg.getDirect() == MsgDirectionEnum.Out) {
                return SEND_VIDEO;
            } else {
                return RECEIVE_VIDEO;
            }
        }
        if (msgType == MsgTypeEnum.location) {
            if (msg.getDirect() == MsgDirectionEnum.Out) {
                return SEND_LOCATION;
            } else {
                return RECEIVE_LOCATION;
            }
        }
        if (msgType == MsgTypeEnum.audio) {
            if (msg.getDirect() == MsgDirectionEnum.Out) {
                return SEND_AUDIO;
            } else {
                return RECEIVE_AUDIO;
            }
        }
        if (msgType == MsgTypeEnum.file) {
            if (msg.getDirect() == MsgDirectionEnum.Out) {
                return SEND_FILE;
            } else {
                return RECEIVE_FILE;
            }
        }
        return super.getItemViewType(position);
    }

    /*==================  begin ==================*/
    private void playAudioDelayAndSetPlayNext(IMMessage item) {
        mAudioControl.startPlayAudioDelay(CLICK_TO_PLAY_AUDIO_DELAY, item,
onPlayListener);
        mAudioControl.setPlayNext(true, SessionAdapter.this, item);
    }

    private MessageAudioControl.AudioControlListener onPlayListener = new
BaseAudioControl.AudioControlListener() {
        @Override
```

```
        public void onAudioControllerReady(Playable playable) {
            play();
            LogUtils.sf(" onAudioControllerReady");
        }

        @Override
        public void onEndPlay(Playable playable) {
            stop();
        }

        @Override
        public void updatePlayingProgress(Playable playable, long curPosition) {

        }
    };

    private void play() {
        if (mAnimationView != null && mAnimationView.getBackground() instanceof
AnimationDrawable) {
            AnimationDrawable animation = (AnimationDrawable) mAnimationView.getBackground();
            animation.start();
        }
    }

    private void stop() {
        if (mAnimationView != null && mAnimationView.getBackground() instanceof
AnimationDrawable) {
            AnimationDrawable animation = (AnimationDrawable) mAnimationView.getBackground();
            animation.stop();
            animation.selectDrawable(0);
        }
    }

    /*==================  end ==================*/
    /*==================  begin ==================*/
    public static int getImageMaxEdge() {
        return (int) (165.0 / 320.0 * ScreenUtil.screenWidth);
    }

    public static int getImageMinEdge() {
        return (int) (76.0 / 320.0 * ScreenUtil.screenWidth);
    }
```

```
    //
    protected void setLayoutParams(int width, int height, View... views) {
        for (View view : views) {
            ViewGroup.LayoutParams maskParams = view.getLayoutParams();
            maskParams.width = width;
            maskParams.height = height;
            view.setLayoutParams(maskParams);
        }
    }

    /*==================  end ==================*/
}
```

54:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\App.java

```
package com.lqr.wechat;

import android.app.Activity;
import android.app.Application;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.os.Environment;
import android.os.Handler;
import android.os.Looper;
import android.text.TextUtils;

import com.lqr.emoji.LQRUIKit;
import com.lqr.imagepicker.ImagePicker;
import com.lqr.imagepicker.view.CropImageView;
import com.lqr.wechat.activity.SplashActivity;
import com.lqr.wechat.imageloader.UILImageLoader;
import com.lqr.wechat.model.UserCache;
import com.lqr.wechat.nimsdk.NimAccountSDK;
import com.lqr.wechat.nimsdk.utils.ScreenUtil;
import com.lqr.wechat.nimsdk.utils.StorageUtils;
import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.SDKOptions;
import com.netease.nimlib.sdk.StatusBarNotificationConfig;
import com.netease.nimlib.sdk.auth.LoginInfo;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
```

```
import com.netease.nimlib.sdk.uinfo.UserInfoProvider;
import com.nostra13.universalimageloader.cache.disc.naming.Md5FileNameGenerator;
import com.nostra13.universalimageloader.core.DisplayImageOptions;
import com.nostra13.universalimageloader.core.ImageLoader;
import com.nostra13.universalimageloader.core.ImageLoaderConfiguration;
import com.nostra13.universalimageloader.core.assist.QueueProcessingType;
import com.zhy.http.okhttp.OkHttpUtils;
import com.zhy.http.okhttp.cookie.CookieJarImpl;
import com.zhy.http.okhttp.cookie.store.PersistentCookieStore;

import java.util.LinkedList;
import java.util.List;
import java.util.concurrent.TimeUnit;

import okhttp3.OkHttpClient;

/**
 * @ CSDN_LQR
 * @ Application
 */
public class App extends Application {

    public static List<Activity> activities = new LinkedList<Activity>();

    @Override
    public void onCreate() {
        super.onCreate();

        //
        mContext = getApplicationContext();
        mMainThread = Thread.currentThread();
        mMainThreadId = android.os.Process.myTid();
        mMainLooper = getMainLooper();
        mHandler = new Handler();


        //LQRUIKitImageLoader
//        initImageLoader(getApplicationContext());

        initNim();
        initImagePicker();
        initOkHttp();
```

```
    }

    private void initNim() {
        LQRUIKit.init(mContext);
        // SDK SDK
        NIMClient.init(this, loginInfo(), options());
        StorageUtils.init(mContext, null);
        ScreenUtil.init(mContext);
    }

    private void initOkHttp() {
        CookieJarImpl cookieJar = new CookieJarImpl(new
PersistentCookieStore(getApplicationContext()));
        OkHttpClient okHttpClient = new OkHttpClient.Builder()
                .cookieJar(cookieJar)
//                .addInterceptor(new LoggerInterceptor("TAG"))
                .connectTimeout(10000L, TimeUnit.MILLISECONDS)
                .readTimeout(10000L, TimeUnit.MILLISECONDS)
                //
                .build();

        OkHttpUtils.initClient(okHttpClient);
    }

    /**
     *
     */
    public static void exit() {
        for (Activity activity : activities) {
            activity.finish();
        }
    }

    /**
     *
     */
    public static void restart() {
        final Intent intent =
mContext.getPackageManager().getLaunchIntentForPackage(mContext.getPackageName());
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        mContext.startActivity(intent);
    }
```

```
//
private static Context mContext;//
private static Thread mMainThread;//
private static long mMainThreadId;//id
private static Looper mMainLooper;//
private static Handler mHandler;//Handler

public static Context getmContext() {
    return mContext;
}

public static void setContext(Context mContext) {
    App.mContext = mContext;
}

public static Thread getMainThread() {
    return mMainThread;
}

public static void setMainThread(Thread mMainThread) {
    App.mMainThread = mMainThread;
}

public static long getMainThreadId() {
    return mMainThreadId;
}

public static void setMainThreadId(long mMainThreadId) {
    App.mMainThreadId = mMainThreadId;
}

public static Looper getMainThreadLooper() {
    return mMainLooper;
}

public static void setMainLooper(Looper mMainLooper) {
    App.mMainLooper = mMainLooper;
}

public static Handler getMainHandler() {
    return mHandler;
```

```
    }

    public static void setMainHandler(Handler mHandler) {
        App.mHandler = mHandler;
    }


    /**
     * ImagePicker
     */
    private void initImagePicker() {
        ImagePicker imagePicker = ImagePicker.getInstance();
        imagePicker.setImageLoader(new UILImageLoader());   //
        imagePicker.setShowCamera(true);  //
        imagePicker.setCrop(true);        //
        imagePicker.setSaveRectangle(true); //
        imagePicker.setSelectLimit(9);    //
        imagePicker.setStyle(CropImageView.Style.RECTANGLE);  //
        imagePicker.setFocusWidth(800);   //
        imagePicker.setFocusHeight(800);  //
        imagePicker.setOutPutX(1000);//
        imagePicker.setOutPutY(1000);//
    }

    public static DisplayImageOptions options = new DisplayImageOptions.Builder()//
            .showImageOnLoading(R.mipmap.default_image)         //
            .showImageForEmptyUri(R.mipmap.default_image)       //Uri
            .showImageOnFail(R.mipmap.default_image)            ///
            .cacheInMemory(true)                                //
            .cacheOnDisk(true)                                  //SD
            .build();                                           //

    /**
     * ImageLoader
     *
     * @param context
     */
    public static void initImageLoader(Context context) {

//        File cacheDir =
com.nostra13.universalimageloader.utils.StorageUtils.getOwnCacheDirectory(context,
"CSDN_LQR/cache");
```

```
        ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(context)
//                .memoryCacheExtraOptions(480, 800) // max width, max height
                .threadPriority(Thread.NORM_PRIORITY - 2)
                .denyCacheImageMultipleSizesInMemory()
                .memoryCacheSize(2 * 1024 * 1024)
                .discCacheSize(50 * 1024 * 1024)
//                .discCacheFileCount(10) //
                .discCacheFileNameGenerator(new
Md5FileNameGenerator()).tasksProcessingOrder(QueueProcessingType.LIFO)
//                .discCache(new UnlimitedDiscCache(cacheDir))//
                .defaultDisplayImageOptions(options)//DisplayImageOptions.createSimple()
                .writeDebugLogs()
                .build();

        ImageLoader.getInstance().init(config);
    }


    /*==================  Begin ==================*/
    //  null
    private SDKOptions options() {
        SDKOptions options = new SDKOptions();

        //  SDK
        StatusBarNotificationConfig config = new StatusBarNotificationConfig();
        config.notificationEntrance = SplashActivity.class; // Activity
        config.notificationSmallIconId = R.mipmap.ic_launcher;
        //
        config.ledARGB = Color.GREEN;
        config.ledOnMs = 1000;
        config.ledOffMs = 1500;
        // uri
        config.notificationSound = "android.resource://com.lqr.wechat/raw/msg";
        options.statusBarNotificationConfig = config;

        // log
        //  options SDK  SDK
        //  log, file, image, audio, video, thumb 6
        //  APP
        String sdkPath = Environment.getExternalStorageDirectory() + "/" + getPackageName() +
"/nim";
        options.sdkStorageRootPath = sdkPath;
```

```
    //  true
    options.preloadAttach = true;

    //
    //   Screen.width / 2
    options.thumbnailSize = 720 / 2;

    // ,
    options.userInfoProvider = new UserInfoProvider() {
        @Override
        public UserInfo getUserInfo(String account) {
            return null;
        }

        @Override
        public int getDefaultIconResId() {
            return R.mipmap.avatar_def;
        }

        @Override
        public Bitmap getTeamIcon(String tid) {
            return null;
        }

        @Override
        public Bitmap getAvatarForMessageNotifier(String account) {
            return null;
        }

        @Override
        public String getDisplayNameForMessageNotifier(String account, String sessionId,
                                                       SessionTypeEnum sessionType) {
            return null;
        }
    };
    return options;
}

// LoginInfonull
private LoginInfo loginInfo() {
    //
```

```
        String account = NimAccountSDK.getUserAccount();
        String token = NimAccountSDK.getUserAccount();

        if (!TextUtils.isEmpty(account) && !TextUtils.isEmpty(token)) {
            //
            UserCache.setAccount(account);
            return new LoginInfo(account, token);
        } else {
            return null;
        }
    }

    /*==================  End ==================*/
}
```

55:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\AppConst.java

```
package com.lqr.wechat;


import com.lqr.wechat.utils.LogUtils;

/**
 * @ CSDN_LQR
 * @
 */
public class AppConst {

    public static final String TAG = "CSDN_LQR";
    public static final int DEBUGLEVEL = LogUtils.LEVEL_ALL;//
    public static final int CACHELTIMEOUT = 10 * 60 * 1000;// 10()

    public static final String NETWORK_CHANGE_RECEIVED_ACTION =
"android.net.conn.CONNECTIVITY_CHANGE";
    public static final String SMS_RECEIVED_ACTION =
"android.provider.Telephony.SMS_RECEIVED";

    public static final String SERVER_ADDRESS = "http://xxx.com/client";

    public static final class Account {
        public static final String KEY_USER_ACCOUNT = "account";
        public static final String KEY_USER_TOKEN = "token";
    }
```

```
    //
    public static final class QRCodeCommend {
        public static final String ACCOUNT = "account:";
        public static final String TEAMID = "teamId:";
    }

    //
    public static final class User {
        private static final String USER = SERVER_ADDRESS + "/user";
        public static final String LOGIN = USER + "/login";//
        public static final String REGISTER = USER + "/insertOrUpdate";//
        public static final String WX_LOGIN = USER + "/androidWXLogin";//
    }

    public static final class Url {
        //
        public static final String HELP_FEEDBACK =
"https://kf.qq.com/touch/product/wechat_app.html?scene_id=kf338&code=021njRdi0RdQfk1Khybi
0kEQdi0njRde&state=123";
        //
        public static final String SHOP =
"http://wqs.jd.com/portal/wx/portal_indexV4.shtml?PTAG=17007.13.1&ptype=1";
        //
        public static final String GAME = "http://h.4399.com/android";
    }

    //
    public static final class UserInfoExt {
        public static final String AREA = "area";
        public static final String PHONE = "phone";
    }

    //
    public static final class MyTeamMemberExt {
        public static final String SHOULD_SHOW_NICK_NAME = "shouldShowNickName";
    }

}
```

56:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\broadcast\AuthBroadcastReceiver.java

```
package com.lqr.wechat.broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

import com.lqr.wechat.activity.LoginActivity;
import com.lqr.wechat.utils.UIUtils;

import static com.netease.nimlib.sdk.StatusCode.*;

/**
 * @ CSDN_LQR
 * @
 */
public class AuthBroadcastReceiver extends BroadcastReceiver {

    public static String ACTION = AuthBroadcastReceiver.class.getName();

    @Override
    public void onReceive(Context context, Intent intent) {

        if (intent.getAction().equals(ACTION)) {

            int status = intent.getIntExtra("status", -1);
            if (status == FORBIDDEN.getValue()) {
                UIUtils.showToast("");
            } else if (status == KICKOUT.getValue()) {
                UIUtils.showToast("");
            } else if (status == KICK_BY_OTHER_CLIENT.getValue()) {
                UIUtils.showToast("");
            } else if (status == PWD_ERROR.getValue()) {
                UIUtils.showToast("");
            }
            Intent i = new Intent(context, LoginActivity.class);
            i.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(i);

        }

    }
}
```

57:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\broadcast\NetWorkChangeBroadcastReceiver.java

```
package com.lqr.wechat.broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

import com.lqr.wechat.AppConst;


/**
 * @ CSDN_LQR
 * @
 */
public class NetWorkChangeBroadcastReceiver extends BroadcastReceiver {
    private NetWorkChangeListener mNetWorkChangeListener;

    public NetWorkChangeBroadcastReceiver(NetWorkChangeListener netWorkChangeListener) {
        super();
        this.mNetWorkChangeListener = netWorkChangeListener;
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(AppConst.NETWORK_CHANGE_RECEIVED_ACTION)) {
            ConnectivityManager connectivityManager =
                    (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
            if (connectivityManager != null) {
                NetworkInfo[] networkInfos = connectivityManager.getAllNetworkInfo();
                for (int i = 0; i < networkInfos.length; i++) {
                    NetworkInfo.State state = networkInfos[i].getState();
                    //
                    if (NetworkInfo.State.CONNECTED == state) {
                        mNetWorkChangeListener.onReceived();
                        return;
                    }
                }
```

```
            }
        }
    }

    /**
     *
     */
    public interface NetWorkChangeListener {
        void onReceived();
    }

}
```

58:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\broadcast\SMSBroadcastReceiver.java

```
package com.lqr.wechat.broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.telephony.SmsMessage;

import com.lqr.wechat.AppConst;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * @ CSDN_LQR
 * @
 */
public class SMSBroadcastReceiver extends BroadcastReceiver {
    private static MessageListener mMessageListener;

    public SMSBroadcastReceiver(MessageListener messageListener) {
        super();
        mMessageListener = messageListener;
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(AppConst.SMS_RECEIVED_ACTION)) {
```

```
            Object[] pdus = (Object[]) intent.getExtras().get("pdus");
            for (Object pdu : pdus) {
                SmsMessage smsMessage = SmsMessage.createFromPdu((byte[]) pdu);
                //
                String content = smsMessage.getDisplayMessageBody();

                //
                int a = content.indexOf("");
                if (a > 0) {
                    //
                    Pattern p = Pattern.compile("\\d{6}");
                    Matcher m = p.matcher(content);
                    m.find();
                    String code = m.group();
                    if (code != null && !code.equals("")) {
                        mMessageListener.onReceived(code);
                        //
                        abortBroadcast();
                    }
                }
            }
        }

    }

    //
    public interface MessageListener {
        void onReceived(String message);
    }

}
```

59:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\factory\ListViewFactory.java

```
package com.lqr.wechat.factory;

import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.widget.ListView;

import com.lqr.wechat.utils.UIUtils;
```

```
/**
 * @ CSDN_LQR
 * @ ListView
 */
public class ListViewFactory {

    public static ListView createListView() {
        ListView listView = new ListView(UIUtils.getContext());

        //
        listView.setCacheColorHint(Color.TRANSPARENT);
        listView.setFastScrollEnabled(true);

        //listviewitem
        listView.setSelector(new ColorDrawable(Color.TRANSPARENT));
        return listView;
    }
}
```

60:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\factory\PopupWindowFactory.java

```
package com.lqr.wechat.factory;

import android.annotation.TargetApi;
import android.app.Activity;
import android.graphics.drawable.BitmapDrawable;
import android.os.Build;
import android.support.annotation.NonNull;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.PopupWindow;

/**
 * @
 * @ 2016/8/19 0019
 * @ popupwindow
 */
public class PopupWindowFactory {
```

```
    /**
     * popupwindow
     *
     * @param contentView popupwindow
     * @param parentView
     * @return
     */
    public static PopupWindow getPopupWindowInCenter(View contentView, View parentView) {
        //        int width = ViewGroup.LayoutParams.MATCH_PARENT;
        int width = ViewGroup.LayoutParams.WRAP_CONTENT;
        int height = ViewGroup.LayoutParams.WRAP_CONTENT;

        return getPopupWindowInCenter(contentView, width, height, parentView);
    }

    /**
     * popupwindow
     *
     * @param contentView popupwindow
     * @param width       popupwindow
     * @param height      popupwindow
     * @param parentView
     * @return
     */
    public static PopupWindow getPopupWindowInCenter(View contentView, int width, int height,
View parentView) {
        //Gravity.CENTER:
        return getPopupWindowAtLocation(contentView, width, height, parentView, Gravity.CENTER,
0, 0);
    }

    /**
     * popupwindow
     *
     * @param contentView popupwindow
     * @param width       popupwindow
     * @param height      popupwindow
     * @param parentView
     * @param gravityType
     * @param xoff        x
     * @param yoff        y
     * @return
```

```
     */
    public static PopupWindow getPopupWindowAtLocation(View contentView, int width, int height,
View parentView, int gravityType, int xoff, int yoff) {
        PopupWindow popupWindow = getPopupWindow(contentView, width, height);

        //parentViewxoffyoff
        popupWindow.showAtLocation(parentView,
                gravityType, xoff, yoff);

        return popupWindow;
    }

    public static PopupWindow getPopupWindowAtLocation(View contentView, View parentView,
int gravityType, int xoff, int yoff) {
        return getPopupWindowAtLocation(contentView,
ViewGroup.LayoutParams.WRAP_CONTENT, ViewGroup.LayoutParams.WRAP_CONTENT,
parentView, gravityType, xoff, yoff);
    }



    /**
     * pupupwindow
     *
     * @param contentView popupwindow
     * @param width       popupwindow
     * @param activity    getWindowManager()
     * @return
     */
    public static PopupWindow getPopupWindowAsDropDownParentAuto(View contentView, int
width, int height, View anchorView, Activity activity) {

        //        View itemView = (View) contentView.getParent();// contentView
        PopupWindow popupWindow = getPopupWindow(contentView, width, height);

        //
        if (isShowBottom(activity, anchorView)) {// popupwindowitemView0
            popupWindow.showAsDropDown(anchorView, 0, 0);
        } else {// popupwindowitemViewy-2*itemView.getHeight()
            popupWindow.showAsDropDown(anchorView, 0,
                    -2 * anchorView.getHeight());
        }
```

```
        return popupWindow;
    }


    /**
     * popupwindow
     *
     * @param contentView popupwindow
     * @param width       popupwindow
     * @param height      popupwindow
     * @param anchorView
     * @param xoff        x
     * @param yoff        y
     * @return
     */
    public static PopupWindow getPopupWindowAsDropDown(View contentView, int width, int
height, View anchorView, int xoff, int yoff) {
        PopupWindow popupWindow = getPopupWindow(contentView, width, height);
        popupWindow.showAsDropDown(anchorView, xoff, yoff);
        return popupWindow;
    }

    /**
     * popupwindow(4.4)
     *
     * @param contentView popupwindow
     * @param width       popupwindow
     * @param height      popupwindow
     * @param anchorView
     * @param gravityType
     * @param xoff        x
     * @param yoff        y
     * @return
     */
    @TargetApi(Build.VERSION_CODES.KITKAT)
    public static PopupWindow getPopupWindowAsDropDown(View contentView, int width, int
height, View anchorView, int gravityType, int xoff, int yoff) {
        PopupWindow popupWindow = getPopupWindow(contentView, width, height);
        popupWindow.showAsDropDown(anchorView, xoff, yoff, gravityType);
        return popupWindow;
    }
```

```
/**
 * popupWindow
 *
 * @param itemView
 * @return
 */
private static boolean isShowBottom(Activity context, View itemView) {
    //
    // int heightPixels =
    // getResources().getDisplayMetrics().heightPixels;//1
    int screenHeight = context.getWindowManager().getDefaultDisplay().getHeight();// 2

    int[] location = new int[2];
    // location[0]-->x
    // location[1]-->y
    itemView.getLocationInWindow(location);
    // itemViewY
    int itemViewY = location[1];

    // itemView
    int distance = screenHeight - itemViewY - itemView.getHeight();

    if (distance < itemView.getHeight()) {// popupWindow
        return false;
    } else {// popupWindow
        return true;
    }
}

/**
 * pupupwindow
 *
 * @param contentView popupwindow
 * @param width       popupwindow
 * @param height      popupwindow
 * @return
 */
@NonNull
private static PopupWindow getPopupWindow(View contentView, int width, int height) {
    PopupWindow popupWindow = new PopupWindow(contentView, width, height, true);
    popupWindow.setOutsideTouchable(false);
```

```
        openOutsideTouchable(popupWindow);
        return popupWindow;
    }

    /**
     * popupwindow
     *
     * @param popupWindow
     */
    public static void openOutsideTouchable(PopupWindow popupWindow) {
        popupWindow.setBackgroundDrawable(new BitmapDrawable());
        popupWindow.setOutsideTouchable(true);
    }

    /**
     * window
     */
    public static void makeWindowDark(Activity activity) {
        makeWindowDark(activity, 0.7f);
    }

    public static void makeWindowDark(Activity activity, float alpha) {
        WindowManager.LayoutParams lp = activity.getWindow().getAttributes();
        lp.alpha = alpha;
        activity.getWindow().setAttributes(lp);
    }

    /**
     * window
     */
    public static void makeWindowLight(Activity activity) {
        WindowManager.LayoutParams lp = activity.getWindow().getAttributes();
        lp.alpha = 1f;
        activity.getWindow().setAttributes(lp);
    }


}
```

61:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\factory\ThreadPoolFactory.java

```
package com.lqr.wechat.factory;
```

```
import com.lqr.wechat.manager.ThreadPoolProxy;

/**
 * @ CSDN_LQR
 * @
 */
public class ThreadPoolFactory {

    static ThreadPoolProxy mNormalPool;
    static ThreadPoolProxy mDownLoadPool;

    /**
     *
     *
     * @return
     */
    public static ThreadPoolProxy getNormalPool() {
        if (mNormalPool == null) {
            synchronized (ThreadPoolFactory.class) {
                if (mNormalPool == null) {
                    mNormalPool = new ThreadPoolProxy(5, 5, 3000);
                }
            }
        }
        return mNormalPool;
    }

    /**
     *
     *
     * @return
     */
    public static ThreadPoolProxy getDownLoadPool() {
        if (mDownLoadPool == null) {
            synchronized (ThreadPoolFactory.class) {
                if (mDownLoadPool == null) {
                    mDownLoadPool = new ThreadPoolProxy(3, 3, 3000);
                }
            }
        }
```

```
        return mDownLoadPool;
    }
}
```

62:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\fragment\BaseFragment.java

```
package com.lqr.wechat.fragment;

import android.app.Dialog;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.lqr.wechat.R;
import com.lqr.wechat.view.CustomDialog;

import me.drakeet.materialdialog.MaterialDialog;


/**
 * @ CSDN_LQR
 * @ Fragment
 */
public abstract class BaseFragment extends Fragment {

    private CustomDialog mDialogWaiting;
    private MaterialDialog mMaterialDialog;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        init();
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return initView();
```

```
}

@Override
public void onActivityCreated(Bundle savedInstanceState) {
    initData();
    initListener();
    super.onActivityCreated(savedInstanceState);
}

/**
 * fview
 *
 * @return
 */
public abstract View initView();

public void init() {

}

public void initData() {

}

public void initListener() {

}

/**
 *
 */
public Dialog showWaitingDialog(String tip) {
    hideWaitingDialog();
    View view = View.inflate(getActivity(), R.layout.dialog_waiting, null);
    if (!TextUtils.isEmpty(tip))
        ((TextView) view.findViewById(R.id.tvTip)).setText(tip);
    mDialogWaiting = new CustomDialog(getActivity(), view, R.style.dialog);
    mDialogWaiting.show();
    return mDialogWaiting;
}

/**
```

```
     *
     */
    public void hideWaitingDialog() {
        if (mDialogWaiting != null) {
            mDialogWaiting.dismiss();
            mDialogWaiting = null;
        }
    }

    /**
     * MaterialDialog
     */
    public MaterialDialog showMaterialDialog(String tip, String message, String positiveText, String
negativeText, View.OnClickListener positiveButtonClickListener, View.OnClickListener
negativeButtonClickListener) {
        hideMaterialDialog();
        mMaterialDialog = new MaterialDialog(getActivity());
        if (!TextUtils.isEmpty(tip)) {
            mMaterialDialog.setTitle(tip);
        }
        if (!TextUtils.isEmpty(message)) {
            mMaterialDialog.setMessage(message);
        }
        if (!TextUtils.isEmpty(positiveText)) {
            mMaterialDialog.setPositiveButton(positiveText, positiveButtonClickListener);
        }
        if (!TextUtils.isEmpty(negativeText)) {
            mMaterialDialog.setNegativeButton(negativeText, negativeButtonClickListener);
        }
        mMaterialDialog.show();
        return mMaterialDialog;
    }

    /**
     * MaterialDialog
     */
    public void hideMaterialDialog() {
        if (mMaterialDialog != null) {
            mMaterialDialog.dismiss();
            mMaterialDialog = null;
        }
    }
```

}

63:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\fragment\ContactsFragment.java

```
package com.lqr.wechat.fragment;

import android.content.Intent;
import android.text.TextUtils;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
import com.lqr.recyclerview.LQRRecyclerView;
import com.lqr.wechat.R;
import com.lqr.wechat.activity.AllTagActvitiy;
import com.lqr.wechat.activity.MainActivity;
import com.lqr.wechat.activity.NewFriendActivity;
import com.lqr.wechat.activity.TeamCheatListActivity;
import com.lqr.wechat.activity.UserInfoActivity;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.Contact;
import com.lqr.wechat.nimsdk.NimFriendSDK;
import com.lqr.wechat.nimsdk.NimRecentContactSDK;
import com.lqr.wechat.nimsdk.NimSystemSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.SortUtils;
import com.lqr.wechat.utils.StringUtils;
import com.lqr.wechat.utils.UIUtils;
import com.lqr.wechat.view.QuickIndexBar;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.friend.model.Friend;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.msg.constant.SystemMessageType;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.ArrayList;
```

```
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;

import static com.netease.nimlib.sdk.msg.constant.SystemMessageType.TeamInvite;

/**
 * @ CSDN_LQR
 * @
 */
public class ContactsFragment extends BaseFragment {

    private List<Contact> mContacts = new ArrayList<>();
    private LQRAdapterForRecyclerView<Contact> mAdapter;
    private int i;
    private List<Friend> mFriends = new ArrayList<>();

    @InjectView(R.id.rvContacts)
    LQRRecyclerView mRvContacts;
    @InjectView(R.id.quickIndexBar)
    QuickIndexBar mQuickIndexBar;
    @InjectView(R.id.tvLetter)
    TextView mTvLetter;

    //
    View mHeaderView;
    TextView mFooterTv;

    //
    LinearLayout mLlNewFriend;
    LinearLayout mLlGroupCheat;
    LinearLayout mLlTag;
    LinearLayout mLlOffical;
    private View mVNewFriendUnread;
    private View mVGroupCheatUnread;


    @Override
    public View initView() {
        View view = View.inflate(getActivity(), R.layout.fragment_contacts, null);
        ButterKnife.inject(this, view);
```

```
        initHeaderViewAndFooterView();
        return view;
    }

    @Override
    public void initData() {

        try {
            mFriends.clear();
            mContacts.clear();

            //
            List<Friend> friends = NimFriendSDK.getFriends();
            if (!StringUtils.isEmpty(friends)) {
                mFriends.addAll(friends);

                //
                List<String> accountList = new ArrayList<>();
                for (int i = 0; i < mFriends.size(); i++) {
                    String account = mFriends.get(i).getAccount();
                    if (NimUserInfoSDK.getUser(account) == null) {
                        accountList.add(account);
                    }
                }

                //
                if (!StringUtils.isEmpty(accountList)) {
                    NimUserInfoSDK.getUserInfosFormServer(accountList, new
RequestCallback<List<NimUserInfo>>() {
                        @Override
                        public void onSuccess(List<NimUserInfo> param) {
                            setDataAndUpdateView();
                        }

                        @Override
                        public void onFailed(int code) {
                            UIUtils.showToast("" + code);
                        }

                        @Override
                        public void onException(Throwable exception) {
                            exception.printStackTrace();
```

```
                }
            });
        } else {
            setDataAndUpdateView();
        }
    } else {
        setDataAndUpdateView();
    }
} catch (Exception e) {
    e.printStackTrace();
    initData();
}
}

@Override
public void initListener() {
    mQuickIndexBar.setListener(new QuickIndexBar.OnLetterUpdateListener() {
        @Override
        public void onLetterUpdate(String letter) {
            //
            showLetter(letter);

            //
            if ("".equalsIgnoreCase(letter)) {
                mRvContacts.moveToPosition(0);
            } else if ("".equalsIgnoreCase(letter)) {
                mRvContacts.moveToPosition(0);
            } else {
                //
                for (i = 0; i < mContacts.size(); i++) {
                    Contact contact = mContacts.get(i);
                    String c = contact.getPinyin().charAt(0) + "";
                    if (c.equalsIgnoreCase(letter)) {
                        mRvContacts.moveToPosition(i);
                        break;
                    }
                }
            }
        }
    });

    mLlNewFriend.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                getActivity().startActivityForResult(new Intent(getActivity(), NewFriendActivity.class),
MainActivity.REQ_CLEAR_UNREAD);
            }
        });

        mLlGroupCheat.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getActivity().startActivity(new Intent(getActivity(), TeamCheatListActivity.class));
            }
        });

        mLlTag.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                getActivity().startActivity(new Intent(getActivity(), AllTagActvitiy.class));
            }
        });
        mLlOffical.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                UIUtils.showToast("");
            }
        });

    }

    @Override
    public void onResume() {
        super.onResume();
        updateHeaderViewUnreadCount();
//        ((MainActivity) getActivity()).updateContactCount();
    }

    private void initHeaderViewAndFooterView() {
        mHeaderView = View.inflate(getActivity(), R.layout.header_contacts_rv, null);

        mLlNewFriend = (LinearLayout) mHeaderView.findViewById(R.id.llNewFriend);
        mLlGroupCheat = (LinearLayout) mHeaderView.findViewById(R.id.llGroupCheat);
        mLlTag = (LinearLayout) mHeaderView.findViewById(R.id.llTag);
```

```
        mLlOffical = (LinearLayout) mHeaderView.findViewById(R.id.llOffical);

        mVNewFriendUnread = mHeaderView.findViewById(R.id.vNewFriendUnread);
        mVGroupCheatUnread = mHeaderView.findViewById(R.id.vGroupCheatUnread);

        mFooterTv = new TextView(getContext());
        ViewGroup.LayoutParams params = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, UIUtils.dip2Px(50));
        mFooterTv.setLayoutParams(params);
        mFooterTv.setGravity(Gravity.CENTER);
    }

    /**
     *
     */
    public void updateHeaderViewUnreadCount() {
        List<SystemMessageType> types = new ArrayList<>(1);
        types.add(SystemMessageType.AddFriend);
        int unreadCountAddFriend =
NimSystemSDK.querySystemMessageUnreadCountByType(types);
        mVNewFriendUnread.setVisibility(unreadCountAddFriend > 0 ? View.VISIBLE :
View.GONE);

        types.clear();
        types.add(TeamInvite);
        int unreadCountTeamInvite =
NimSystemSDK.querySystemMessageUnreadCountByType(types);
        mVGroupCheatUnread.setVisibility(unreadCountTeamInvite > 0 ? View.VISIBLE :
View.GONE);
    }

    private void setDataAndUpdateView() {
        if (mFriends != null) {
            for (int i = 0; i < mFriends.size(); i++) {
                Friend friend = mFriends.get(i);
                NimUserInfo userInfo = NimUserInfoSDK.getUser(friend.getAccount());
                mContacts.add(new Contact(friend, userInfo));
            }

            //
//            mContacts.add(new Contact(null, NimUserInfoSDK.getUser(UserCache.getAccount())));
            //
```

```
            SortUtils.sortContacts(mContacts);

            if (mFooterTv != null) {
                mFooterTv.setVisibility(View.VISIBLE);
                mFooterTv.setText(mContacts.size() + "");
            }
        } else {
            mFooterTv.setVisibility(View.GONE);
        }
        setAdapter();
    }


    private void setAdapter() {
        mAdapter = new LQRAdapterForRecyclerView<Contact>(getActivity(),
R.layout.item_contact_cv, mContacts) {
            @Override
            public void convert(LQRViewHolderForRecyclerView helper, final Contact item, int
position) {
                helper.setText(R.id.tvName, TextUtils.isEmpty(item.getAlias()) ? item.getName() :
item.getAlias());
                if (!TextUtils.isEmpty(item.getAvatar())) {
                    ImageLoaderManager.LoadNetImage(item.getAvatar(), (ImageView)
helper.getView(R.id.ivHeader));
                } else {
                    helper.setImageResource(R.id.ivHeader, R.mipmap.default_header);
                }

                String str = "";
                //
                String currentLetter = item.getPinyin().charAt(0) + "";

                if (position == 0) {
                    str = currentLetter;
                } else {
                    //
                    String preLetter = mContacts.get(position - 1).getPinyin().charAt(0) + "";
                    //
                    if (!preLetter.equalsIgnoreCase(currentLetter)) {
                        str = currentLetter;
                    }
```

```
            int nextIndex = position + 1;
            if (nextIndex < mContacts.size() - 1) {
                //
                String nextLetter = mContacts.get(nextIndex).getPinyin().charAt(0) + "";
                //
                if (!nextLetter.equalsIgnoreCase(currentLetter)) {
                    helper.setViewVisibility(R.id.vLine, View.INVISIBLE);
                } else {
                    helper.setViewVisibility(R.id.vLine, View.VISIBLE);
                }
            } else {
                helper.setViewVisibility(R.id.vLine, View.INVISIBLE);
            }
        }
        if (position == mContacts.size() - 1) {
            helper.setViewVisibility(R.id.vLine, View.GONE);
        }


        //str
        if (TextUtils.isEmpty(str)) {
            helper.setViewVisibility(R.id.tvIndex, View.GONE);
        } else {
            helper.setViewVisibility(R.id.tvIndex, View.VISIBLE);
            helper.setText(R.id.tvIndex, currentLetter);
        }

        //
        helper.getView(R.id.root).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getActivity(), UserInfoActivity.class);
                intent.putExtra("account", item.getAccount());
                startActivity(intent);
                //
                NimRecentContactSDK.clearUnreadCount(item.getAccount(),
SessionTypeEnum.P2P);
            }
        });

    }
};
```

```
        //
        mAdapter.addHeaderView(mHeaderView);
        //
        mAdapter.addFooterView(mFooterTv);
        //
        if (mRvContacts != null)
            mRvContacts.setAdapter(mAdapter.getHeaderAndFooterAdapter());
    }

    /**
     *
     *
     * @param letter
     */
    protected void showLetter(String letter) {
        mTvLetter.setVisibility(View.VISIBLE);//
        mTvLetter.setText(letter);

        UIUtils.getMainThreadHandler().removeCallbacksAndMessages(null);
        UIUtils.postTaskDelay(new Runnable() {
            @Override
            public void run() {
                mTvLetter.setVisibility(View.GONE);
            }
        }, 500);
    }

    /**
     *
     *
     * @param show
     */
    public void showQuickIndexBar(boolean show) {
        mQuickIndexBar.setVisibility(show ? View.VISIBLE : View.GONE);
        mQuickIndexBar.invalidate();
    }
}
```

64:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\fragment\DiscoveryFragment.java

```
package com.lqr.wechat.fragment;
```

```
import android.content.Intent;
import android.view.View;

import com.lqr.wechat.AppConst;
import com.lqr.wechat.R;
import com.lqr.wechat.activity.NearbyPerpleActivity;
import com.lqr.wechat.activity.ScanActivity;
import com.lqr.wechat.activity.WebViewActivity;

import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * @ CSDN_LQR
 * @
 */
public class DiscoveryFragment extends BaseFragment {

    private Intent mIntent;

    @OnClick({R.id.oivScan, R.id.oivNearby, R.id.oivShop, R.id.oivGame})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.oivScan:
                startActivity(new Intent(getActivity(), ScanActivity.class));
                break;
            case R.id.oivNearby:
                startActivity(new Intent(getActivity(), NearbyPerpleActivity.class));
                break;
            case R.id.oivShop:
                mIntent = new Intent(getActivity(), WebViewActivity.class);
                mIntent.putExtra("url", AppConst.Url.SHOP);
                mIntent.putExtra("title", "");
                startActivity(mIntent);
                break;
            case R.id.oivGame:
                mIntent = new Intent(getActivity(), WebViewActivity.class);
                mIntent.putExtra("url", AppConst.Url.GAME);
                mIntent.putExtra("title", "");
                startActivity(mIntent);
                break;
        }
```

```
    }

    @Override
    public View initView() {
        View view = View.inflate(getActivity(), R.layout.fragment_discovery, null);
        ButterKnife.inject(this, view);
        return view;
    }
}
```

65:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\fragment\Func1Fragment.java

```
package com.lqr.wechat.fragment;

import android.content.Intent;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.lqr.imagepicker.ui.ImageGridActivity;
import com.lqr.wechat.R;
import com.lqr.wechat.activity.RedPacketActivity;
import com.lqr.wechat.activity.SessionActivity;
import com.lqr.wechat.activity.TransferActivity;
import com.lqr.wechat.view.CustomDialog;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;

import static com.lqr.wechat.R.id.tvOne;
import static com.lqr.wechat.R.id.tvTwo;
import static com.lqr.wechat.activity.SessionActivity.IMAGE_PICKER;

/**
 * @ CSDN_LQR
 * @ 1
 */
public class Func1Fragment extends BaseFragment {

    private View mContentView;
    private CustomDialog mDialog;
```

```
    private TextView mTvOne;
    private TextView mTvTwo;

    @InjectView(R.id.llPic)
    LinearLayout mLlPic;
    @InjectView(R.id.llRecord)
    LinearLayout mLlRecord;
    @InjectView(R.id.llRedPacket)
    LinearLayout mLlRedPacket;
    @InjectView(R.id.llTransfer)
    LinearLayout mLlTransfer;

    @InjectView(R.id.llCollection)
    LinearLayout mLlCollection;
    @InjectView(R.id.llLocation)
    LinearLayout mLlLocation;
    @InjectView(R.id.llVideo)
    LinearLayout mLlVideo;
    @InjectView(R.id.llBusinessCard)
    LinearLayout mLlBusinessCard;

    Intent mIntent;

    @OnClick({R.id.llPic, R.id.llRecord, R.id.llRedPacket, R.id.llTransfer, R.id.llLocation,
R.id.llVideo})
    public void click(View view) {
        switch (view.getId()) {
            case R.id.llPic:
                mIntent = new Intent(getActivity(), ImageGridActivity.class);
                startActivityForResult(mIntent, IMAGE_PICKER);
                break;
            case R.id.llRecord:
                ((SessionActivity)getActivity()).showPlayVideo();
                break;
            case R.id.llRedPacket:
                mIntent = new Intent(getActivity(), RedPacketActivity.class);
                startActivity(mIntent);
                break;
            case R.id.llTransfer:
                mIntent = new Intent(getActivity(), TransferActivity.class);
                startActivity(mIntent);
                break;
```

```
case R.id.llLocation:
    mContentView = View.inflate(getActivity(), R.layout.dialog_menu_two_session, null);
    mDialog = new CustomDialog(getActivity(), mContentView, R.style.dialog);
    mDialog.show();
    mTvOne = (TextView) mContentView.findViewById(tvOne);
    mTvTwo = (TextView) mContentView.findViewById(tvTwo);
    mTvOne.setText("");
    mTvTwo.setText("");
    mTvOne.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mDialog.dismiss();
        }
    });
    mTvTwo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mDialog.dismiss();
        }
    });

    break;
case R.id.llVideo:
    mContentView = View.inflate(getActivity(), R.layout.dialog_menu_two_session, null);
    mDialog = new CustomDialog(getActivity(), mContentView, R.style.dialog);
    mDialog.show();
    mTvOne = (TextView) mContentView.findViewById(tvOne);
    mTvTwo = (TextView) mContentView.findViewById(tvTwo);
    mTvOne.setText("");
    mTvTwo.setText("");
    mTvOne.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mDialog.dismiss();
        }
    });
    mTvTwo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mDialog.dismiss();
        }
    });
```

```
                break;

        }
    }

    @Override
    public View initView() {
        View view = View.inflate(getActivity(), R.layout.fragment_func_page1, null);
        ButterKnife.inject(this, view);
        return view;
    }

}
```

66:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\fragment\Func2Fragment.java

```
package com.lqr.wechat.fragment;

import android.view.View;
import android.widget.LinearLayout;

import com.lqr.wechat.R;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @ 2
 */
public class Func2Fragment extends BaseFragment {

    @InjectView(R.id.llVoice)
    LinearLayout mLlVoice;

    @Override
    public View initView() {
        View view = View.inflate(getActivity(), R.layout.fragment_func_page2, null);
        ButterKnife.inject(this, view);
        return view;
    }
}
```

67:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\fragment\MeFragment.java

```
package com.lqr.wechat.fragment;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.drawable.BitmapDrawable;
import android.text.TextUtils;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import com.lqr.wechat.AppConst;
import com.lqr.wechat.R;
import com.lqr.wechat.activity.CardPaketActivity;
import com.lqr.wechat.activity.MyInfoActivity;
import com.lqr.wechat.activity.SettingActivity;
import com.lqr.wechat.factory.ThreadPoolFactory;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.UserCache;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.UIUtils;
import com.lqr.wechat.view.CustomDialog;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.uinfo.constant.GenderEnum;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;
import cn.bingoogolapple.qrcode.zxing.QRCodeEncoder;

/**
 * @ CSDN_LQR
 * @
 */
public class MeFragment extends BaseFragment {
```

```
    private NimUserInfo mNimUserInfo;
    private View mQRCodeCardView;
    private CustomDialog mQRCodeCardDialog;
    private ImageView mIvHeaderQRCodeCard;
    private TextView mTvNameQRCodeCard;
    private ImageView mIvGenderQRCodeCard;
    private ImageView mIvCardQRCodeCard;

    @InjectView(R.id.ivHeader)
    ImageView mIvHeader;
    @InjectView(R.id.tvName)
    TextView mTvName;
    @InjectView(R.id.tvAccount)
    TextView mTvAccount;


    @OnClick({R.id.llMyInfo, R.id.ivQRCordCard, R.id.oivCardPaket, R.id.oivSetting})
    public void click(View view) {
        Intent intent;
        switch (view.getId()) {
            case R.id.llMyInfo:
                intent = new Intent(getActivity(), MyInfoActivity.class);
                startActivity(intent);
                break;
            case R.id.ivQRCordCard:
                if (mQRCodeCardView == null) {
                    mQRCodeCardView = View.inflate(getActivity(), R.layout.include_qrcode_card, null);
mQRCodeCardView.setBackgroundResource(R.drawable.shape_corner_rect_solid_white);
        mIvHeaderQRCodeCard = (ImageView) mQRCodeCardView.findViewById(R.id.ivHeader);
                    mTvNameQRCodeCard = (TextView)
mQRCodeCardView.findViewById(R.id.tvName);
                    mIvGenderQRCodeCard = (ImageView)
mQRCodeCardView.findViewById(R.id.ivGender);
                    mIvCardQRCodeCard = (ImageView) mQRCodeCardView.findViewById(R.id.ivCard);
                    mQRCodeCardDialog = new CustomDialog(getActivity(), 300, 400,
mQRCodeCardView, R.style.dialog);
                }

                String avatar = mNimUserInfo.getAvatar();
                if (!TextUtils.isEmpty(avatar))
                    ImageLoaderManager.LoadNetImage(avatar, mIvHeaderQRCodeCard);
                else
```

```
                mIvHeaderQRCodeCard.setImageResource(R.mipmap.default_header);
            mTvNameQRCodeCard.setText(mNimUserInfo.getName());
            if (mNimUserInfo.getGenderEnum() == GenderEnum.FEMALE) {
                mIvGenderQRCodeCard.setImageResource(R.mipmap.ic_gender_female);
            } else if (mNimUserInfo.getGenderEnum() == GenderEnum.MALE) {
                mIvGenderQRCodeCard.setImageResource(R.mipmap.ic_gender_male);
            } else {
                mIvGenderQRCodeCard.setVisibility(View.GONE);
            }
            Bitmap bitmap = ((BitmapDrawable) mIvHeader.getDrawable()).getBitmap();
            showQRCordCard(bitmap);
            mQRCodeCardDialog.show();
            break;
        case R.id.oivCardPaket:
            intent = new Intent(getActivity(), CardPaketActivity.class);
            startActivity(intent);
            break;
        case R.id.oivSetting:
            intent = new Intent(getActivity(), SettingActivity.class);
            startActivity(intent);
            break;
    }
}

@Override
public View initView() {
    View view = View.inflate(getActivity(), R.layout.fragment_me, null);
    ButterKnife.inject(this, view);
    return view;
}

@Override
public void initData() {
    mNimUserInfo = NimUserInfoSDK.getUser(UserCache.getAccount());
    if (mNimUserInfo == null) {
        getUserInfoFromRemote();
    } else {
        //
        if (!TextUtils.isEmpty(mNimUserInfo.getAvatar()) && mIvHeader != null) {
            ImageLoaderManager.LoadNetImage(mNimUserInfo.getAvatar(), mIvHeader);
        }
        //
```

```
            if (mTvName != null)
                mTvName.setText(mNimUserInfo.getName());
            if (mTvAccount != null)
                mTvAccount.setText(mNimUserInfo.getAccount());
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mQRCodeCardDialog != null)
            mQRCodeCardDialog.dismiss();
    }

    private void getUserInfoFromRemote() {
        List<String> accountList = new ArrayList<>();
        accountList.add(UserCache.getAccount());
        NimUserInfoSDK.getUserInfosFormServer(accountList, new
RequestCallback<List<NimUserInfo>>() {
            @Override
            public void onSuccess(List<NimUserInfo> param) {
                initData();
            }

            @Override
            public void onFailed(int code) {
                UIUtils.showToast("" + code);
            }

            @Override
            public void onException(Throwable exception) {
                exception.printStackTrace();
            }
        });
    }

    private void showQRCordCard(final Bitmap bitmap) {
        ThreadPoolFactory.getNormalPool().execute(new Runnable() {
            @Override
            public void run() {
//                final Bitmap codeWithLogo5 =
QRCodeEncoder.syncEncodeQRCode(AppConst.QRCodeCommend.ACCOUNT +
```

```
mNimUserInfo.getAccount(), UIUtils.dip2Px(200), UIUtils.getColor(R.color.transparent),
UIUtils.getColor(R.color.black0), bitmap);
            final Bitmap codeWithLogo5 =
QRCodeEncoder.syncEncodeQRCode(AppConst.QRCodeCommend.ACCOUNT +
mNimUserInfo.getAccount(), UIUtils.dip2Px(200));
            UIUtils.postTaskSafely(new Runnable() {
                @Override
                public void run() {
                    mIvCardQRCodeCard.setImageBitmap(codeWithLogo5);
                }
            });
        }
    });
  }
}
```

68:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\fragment\MessageFragment.java

```
package com.lqr.wechat.fragment;

import android.content.Context;
import android.content.Intent;
import android.support.v7.widget.RecyclerView;
import android.text.TextUtils;
import android.text.style.ImageSpan;
import android.view.View;
import android.widget.ImageView;

import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.adapter.LQRViewHolderForRecyclerView;
import com.lqr.emoji.MoonUtil;
import com.lqr.ninegridimageview.LQRNineGridImageView;
import com.lqr.ninegridimageview.LQRNineGridImageViewAdapter;
import com.lqr.wechat.R;
import com.lqr.wechat.activity.MainActivity;
import com.lqr.wechat.activity.SessionActivity;
import com.lqr.wechat.factory.ThreadPoolFactory;
import com.lqr.wechat.imageloader.ImageLoaderManager;
import com.lqr.wechat.model.Contact;
import com.lqr.wechat.nimsdk.NimFriendSDK;
import com.lqr.wechat.nimsdk.NimRecentContactSDK;
import com.lqr.wechat.nimsdk.NimTeamSDK;
```

```
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.TimeUtils;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.ResponseCode;
import com.netease.nimlib.sdk.friend.model.Friend;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.msg.model.RecentContact;
import com.netease.nimlib.sdk.team.model.Team;
import com.netease.nimlib.sdk.team.model.TeamMember;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.util.ArrayList;
import java.util.List;

import butterknife.ButterKnife;
import butterknife.InjectView;

/**
 * @ CSDN_LQR
 * @
 */
public class MessageFragment extends BaseFragment {

    private List<RecentContact> mRecentContactList = new ArrayList<>();//
    private Observer<List<RecentContact>> mMessageObserver;

    private LQRAdapterForRecyclerView<RecentContact> mAdapter;
    private View mHeaderView;
    private LQRNineGridImageViewAdapter<NimUserInfo> mNineGridAdapter;

    @InjectView(R.id.cvMessage)
    RecyclerView mCvMessage;
    MainActivity activity;

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        activity = (MainActivity) context;
    }
```

```
    @Override
    public void onDetach() {
        super.onDetach();
    }

    @Override
    public void init() {
        //
        observeRecentContact();
        //
        updateTotalUnReadCount();
    }

    @Override
    public View initView() {
        View view = View.inflate(getActivity(), R.layout.fragment_message, null);
        ButterKnife.inject(this, view);

//        mHeaderView = View.inflate(getActivity(), R.layout.header_message_rv, null);
//        mHeaderView.setVisibility(View.GONE);

        mNineGridAdapter = new LQRNineGridImageViewAdapter<NimUserInfo>() {
            @Override
            protected void onDisplayImage(Context context, ImageView imageView, NimUserInfo
userInfo) {
                if (!TextUtils.isEmpty(userInfo.getAvatar())) {
                    ImageLoaderManager.LoadNetImage(userInfo.getAvatar(), imageView);
                } else {
                    imageView.setImageResource(R.mipmap.default_header);
                }
            }
        };

        return view;
    }

    @Override
    public void initData() {
        getLocalRecentData();
    }
```

```
//    @Override
//    public void onResume() {
//        super.onResume();
//        setAdapter();
//    }

    private void setAdapter() {
//        if (mAdapter == null) {
            mAdapter = new LQRAdapterForRecyclerView<RecentContact>(getActivity(),
R.layout.item_message_rv, mRecentContactList) {
                @Override
                public void convert(final LQRViewHolderForRecyclerView helper, final RecentContact
item, int position) {
                    final ImageView ivHeader = helper.getView(R.id.ivHeader);//
                    final LQRNineGridImageView ngivHeader = helper.getView(R.id.ngiv);//
                    if (item.getSessionType() == SessionTypeEnum.P2P) {
                        ivHeader.setVisibility(View.VISIBLE);
                        ngivHeader.setVisibility(View.GONE);
                        //
                        Friend friend = NimFriendSDK.getFriendByAccount(item.getContactId());
                        NimUserInfo userInfo = NimUserInfoSDK.getUser(item.getContactId());
                        if (userInfo == null) {
                            return;
                        }
                        Contact contact = new Contact(friend, userInfo);

                        //
                        if (userInfo != null && !TextUtils.isEmpty(userInfo.getAvatar())) {
                            ImageLoaderManager.LoadNetImage(userInfo.getAvatar(), ivHeader);
                        } else {
                            (ivHeader).setImageResource(R.mipmap.default_header);
                        }

                        helper.setText(R.id.tvName, contact.getDisplayName());
                    } else {
                        ivHeader.setVisibility(View.GONE);
                        ngivHeader.setVisibility(View.VISIBLE);
                        ThreadPoolFactory.getNormalPool().execute(new Runnable() {
                            @Override
                            public void run() {
                                final Team team = NimTeamSDK.queryTeamBlock(item.getContactId());
                                if (team == null)
```

```
                return;
            UIUtils.postTaskSafely(new Runnable() {
                @Override
                public void run() {
                    //
                    if (team.isMyTeam()) {
                        if (team != null)
                            helper.setText(R.id.tvName, TextUtils.isEmpty(team.getName()) ? "
(" + team.getMemberCount() + ")" : team.getName());
                    } else {
                        NimRecentContactSDK.deleteRecentContact(item);
                        mAdapter.removeItem(item);
                    }

                    //
                    NimTeamSDK.queryMemberList(team.getId(), new
RequestCallback<List<TeamMember>>() {
                        @Override
                        public void onSuccess(List<TeamMember> memberList) {
                            if (memberList != null && memberList.size() > 0) {
                                List<String> accounts = new ArrayList<>();
                                int count = memberList.size() > 9 ? 9 : memberList.size();
                                for (int i = 0; i < count; i++) {
                                    accounts.add(memberList.get(i).getAccount());
                                }
                                NimUserInfoSDK.getUserInfosFormServer(accounts, new
RequestCallback<List<NimUserInfo>>() {
                                    @Override
                                    public void onSuccess(List<NimUserInfo> result) {
                                        ngivHeader.setAdapter(mNineGridAdapter);
                                        ngivHeader.setImagesData(result);
                                    }

                                    @Override
                                    public void onFailed(int code) {

                                    }

                                    @Override
                                    public void onException(Throwable exception) {

                                    }
```

```
                                    });
                                }
                            }

                            @Override
                            public void onFailed(int code) {

                            }

                            @Override
                            public void onException(Throwable exception) {

                            }
                        });
                    }
                });
            }
        });
    }

    helper.setText(R.id.tvMsg, item.getContent())
            .setText(R.id.tvTime, TimeUtils.getMsgFormatTime(item.getTime()));
//                MoonUtil.identifyFaceExpression(getActivity(), helper.getView(R.id.tvMsg),
item.getContent(), ImageSpan.ALIGN_BOTTOM);
    MoonUtil.identifyFaceExpressionAndTags(getActivity(), helper.getView(R.id.tvMsg),
item.getContent(), ImageSpan.ALIGN_BOTTOM, 0.45f);

    //
    helper.setViewVisibility(R.id.tvUnread, item.getUnreadCount() > 0 ? View.VISIBLE :
View.GONE).setText(R.id.tvUnread, String.valueOf(item.getUnreadCount()));

    //
    helper.getView(R.id.root).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(getActivity(), SessionActivity.class);
            intent.putExtra(SessionActivity.SESSION_ACCOUNT, item.getContactId());
            intent.putExtra(SessionActivity.SESSION_TYPE, item.getSessionType());
            startActivity(intent);
            //
            NimRecentContactSDK.clearUnreadCount(item.getContactId(),
item.getSessionType());
```

```
                    }
                });
            }
        };
//          mAdapter.addHeaderView(mHeaderView);
//          mCvMessage.setAdapter(mAdapter.getHeaderAndFooterAdapter());
        mCvMessage.setAdapter(mAdapter);
//      } else {
//          mAdapter.notifyDataSetChanged();
//      }
    }

    /**
     *
     */
    private void getLocalRecentData() {
        //
        NimRecentContactSDK.queryRecentContacts(new
RequestCallbackWrapper<List<RecentContact>>() {
            @Override
            public void onResult(int code, List<RecentContact> result, Throwable exception) {
                if (code != ResponseCode.RES_SUCCESS || exception != null)
                    return;

                //
                for (int i = 0; i < result.size(); i++) {
                    RecentContact rc = result.get(i);
                    if (rc.getSessionType() == SessionTypeEnum.Team) {
                        if (!NimTeamSDK.queryTeamBlock(rc.getContactId()).isMyTeam()) {
                            result.remove(i);
                            NimRecentContactSDK.deleteRecentContact(rc);
                            i--;
                        }
                    }
                }

                mRecentContactList.clear();
                mRecentContactList.addAll(result);
                setAdapter();

                updateRecentContactInfoFromServer();
```

```
        }
    });
}

/**
 *
 */
private void updateRecentContactInfoFromServer() {
    if (mRecentContactList != null && mRecentContactList.size() > 0) {
        List<String> accounts = new ArrayList<>();
        for (RecentContact rc : mRecentContactList) {
            accounts.add(rc.getFromAccount());
        }
        if (accounts != null && accounts.size() > 0) {
            NimUserInfoSDK.getUserInfosFormServer(accounts, new
RequestCallback<List<NimUserInfo>>() {
                @Override
                public void onSuccess(List<NimUserInfo> param) {
                    setAdapter();
                }

                @Override
                public void onFailed(int code) {

                }

                @Override
                public void onException(Throwable exception) {

                }
            });
        }
    }
}

/**
 *
 */
private void observeRecentContact() {
    mMessageObserver = new Observer<List<RecentContact>>() {
        @Override
        public void onEvent(List<RecentContact> recentContacts) {
```

```
            //
            if (recentContacts != null && recentContacts.size() > 0) {
                if (mAdapter != null) {
                    int index;
                    for (RecentContact r : recentContacts) {
                        index = -1;
                        for (int i = 0; i < mAdapter.getData().size(); i++) {
                            if (r.getContactId().equals(mAdapter.getData().get(i).getContactId())
                                    && r.getSessionType() == (mAdapter.getData().get(i).getSessionType()))
{
                                index = i;
                                break;
                            }
                        }
                        if (index >= 0) {
                            mAdapter.removeItem(index);
                        }
                        mAdapter.addFirstItem(r);
                    }
                    updateTotalUnReadCount();
                }
            }
        };

        NimRecentContactSDK.observeRecentContact(mMessageObserver, true);
    }

    /**
     *
     */
    private void updateTotalUnReadCount() {
        int totalUnreadCount = NimRecentContactSDK.getTotalUnreadCount();
        if (activity.mTvMessageCount != null)
            if (totalUnreadCount > 0) {
                activity.mTvMessageCount.setVisibility(View.VISIBLE);
                activity.mTvMessageCount.setText(String.valueOf(totalUnreadCount > 99 ? 99 :
totalUnreadCount));
            } else {
                activity.mTvMessageCount.setVisibility(View.GONE);
            }
```

```
    }
}
```

69:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\imageloader\ImageLoaderManager.java

```
package com.lqr.wechat.imageloader;

import android.net.Uri;
import android.widget.ImageView;

import com.lqr.wechat.App;

/**
 * @ CSDN_LQR
 * @ (universalimage)
 */
public class ImageLoaderManager {

    public static void LoadNetImage(String imgUrl, ImageView imageView) {
        com.nostra13.universalimageloader.core.ImageLoader.getInstance().displayImage(imgUrl, imageView, App.options);
    }

    public static void LoadLocalImage(String path, ImageView imageView) {
com.nostra13.universalimageloader.core.ImageLoader.getInstance().displayImage(Uri.parse("file:///" + path).toString(), imageView, App.options);
    }


//    public static void LoadNetImage(String imgUrl, ImageView imageView) {
//        com.nostra13.universalimageloader.core.ImageLoader.getInstance().displayImage(imgUrl, imageView);
//    }
//
//    public static void LoadNetImage(String imgUrl, ImageView imageView, DisplayImageOptions o) {
//        com.nostra13.universalimageloader.core.ImageLoader.getInstance().displayImage(imgUrl, imageView, o);
//    }
//
//    public static void LoadLocalImage(String path, ImageView imageView) {
//
```

```
com.nostra13.universalimageloader.core.ImageLoader.getInstance().displayImage(Uri.parse("file:/
/" + path).toString(), imageView);
//    }
//
//    public static void LoadLocalImage(String path, ImageView imageView, DisplayImageOptions
o) {
//
com.nostra13.universalimageloader.core.ImageLoader.getInstance().displayImage(Uri.parse("file:/
/" + path).toString(), imageView, o);
//    }
}
```

70:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\imageloader\UILImage
Loader.java

```
package com.lqr.wechat.imageloader;

import android.app.Activity;
import android.net.Uri;
import android.widget.ImageView;

import com.lqr.imagepicker.loader.ImageLoader;
import com.nostra13.universalimageloader.core.assist.ImageSize;

/**
 * @ CSDN_LQR
 * @
 */
public class UILImageLoader implements ImageLoader {

//    @Override
//    public void displayImage(Activity activity, String path, ImageView imageView, int width, int
height) {
//        ImageLoaderManager.LoadNetImage(Uri.fromFile(new File(path)).toString(), imageView,
App.options);
//    }

    @Override
    public void displayImage(Activity activity, String path, ImageView imageView, int width, int
height) {
        ImageSize size = new ImageSize(width, height);
//
com.nostra13.universalimageloader.core.ImageLoader.getInstance().displayImage(Uri.fromFile(ne
```

```
w File(path)).toString(), imageView, size);
com.nostra13.universalimageloader.core.ImageLoader.getInstance().displayImage(Uri.parse("file:/
/"+path).toString(), imageView, size);
    }

    @Override
    public void clearMemoryCache() {
    }
}
```

71:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\manager\ThreadPoolP
roxy.java

```
package com.lqr.wechat.manager;

import java.util.concurrent.BlockingQueue;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.LinkedBlockingDeque;
import java.util.concurrent.RejectedExecutionHandler;
import java.util.concurrent.ThreadFactory;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;

/**
 * @ CSDN_LQR
 * @
 */
public class ThreadPoolProxy {
    ThreadPoolExecutor mExecutor;//
    int mCorePoolSize;
    int mMaximumPoolSize;
    long mKeepAliveTime;

    public ThreadPoolProxy(int corePoolSize, int maximumPoolSize,
                           long keepAliveTime) {
        super();
        mCorePoolSize = corePoolSize;
        mMaximumPoolSize = maximumPoolSize;
        mKeepAliveTime = keepAliveTime;
    }

    private ThreadPoolExecutor initThreadPoolExecutor() {//
```

```
    if (mExecutor == null) {
        synchronized (ThreadPoolProxy.class) {
            if (mExecutor == null) {
                TimeUnit unit = TimeUnit.MILLISECONDS;//
                BlockingQueue<Runnable> workQueue = new LinkedBlockingDeque<Runnable>();//
                ThreadFactory threadFactory = Executors
                        .defaultThreadFactory();
                RejectedExecutionHandler handler = new ThreadPoolExecutor.AbortPolicy();//

                mExecutor = new ThreadPoolExecutor(//
                        mCorePoolSize, //
                        mMaximumPoolSize,//
                        mKeepAliveTime, //
                        unit, //
                        workQueue,// /
                        threadFactory,//
                        handler//
                );
            }
        }

    }
    return mExecutor;
}

/**
 *
 *
 * @param task
 */
public void execute(Runnable task) {
    initThreadPoolExecutor();
    mExecutor.execute(task);
}

/**
 *
 *
 * @param task
 */
public Future<?> submit(Runnable task) {
    initThreadPoolExecutor();
```

```
        return mExecutor.submit(task);
    }

    /**
     *
     *
     * @param task
     */
    public void removeTask(Runnable task) {
        initThreadPoolExecutor();
        mExecutor.remove(task);
    }
}
```

72:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\model\Contact.java

```
package com.lqr.wechat.model;

import android.text.TextUtils;

import com.lqr.wechat.nimsdk.NimFriendSDK;
import com.lqr.wechat.nimsdk.NimUserInfoSDK;
import com.lqr.wechat.utils.PinyinUtils;
import com.netease.nimlib.sdk.friend.model.Friend;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.io.Serializable;

/**
 * @ CSDN_LQR
 * @
 */
public class Contact implements Comparable<Contact>, Serializable {

    private String mAccount;//
    private String mDisplayName;//
    private String mName;//
    private String mAlias;//
    private String mPinyin;///
    private Friend mFriend;//
    private NimUserInfo mUserInfo;//
    private String mAvatar;//
//    private List<String> mAccounts;//
```

```
public Contact(Friend friend, NimUserInfo userInfo) {
    mFriend = friend;
    mUserInfo = userInfo;
    fit();
}

public Contact(String account) {
    super();
    mFriend = NimFriendSDK.getFriendByAccount(account);
    mUserInfo = NimUserInfoSDK.getUser(account);
    fit();
}

public Contact() {
    super();
}

private void fit() {
    this.mAccount = mUserInfo.getAccount();
    this.mName = mUserInfo.getName();
    if (mFriend != null)
        this.mAlias = mFriend.getAlias();
    this.mAvatar = mUserInfo.getAvatar();
    this.mDisplayName = TextUtils.isEmpty(mAlias) ? mName : mAlias;
    this.mPinyin = PinyinUtils.getPinyin(mDisplayName);
}

public String getAccount() {
    return mAccount;
}

public void setAccount(String account) {
    mAccount = account;
}

public String getAlias() {
    return mAlias;
}

public void setAlias(String alias) {
    this.mAlias = alias;
```

```
}

public String getPinyin() {
    return mPinyin;
}

public void setPinyin(String pinyin) {
    this.mPinyin = pinyin;
}

public Friend getFriend() {
    return mFriend;
}

public void setFriend(Friend friend) {
    mFriend = friend;
}

public NimUserInfo getUserInfo() {
    return mUserInfo;
}

public void setUserInfo(NimUserInfo userInfo) {
    mUserInfo = userInfo;
}

public String getAvatar() {
    return mAvatar;
}

public void setAvatar(String avatar) {
    mAvatar = avatar;
}

public String getName() {
    return mName;
}

public void setName(String name) {
    mName = name;
}
```

```
    public String getDisplayName() {
        return mDisplayName;
    }

    public void setDisplayName(String displayName) {
        mDisplayName = displayName;
    }

    @Override
    public int compareTo(Contact o) {
        return this.mPinyin.compareTo(o.getPinyin());
    }
}
```

73:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\model\NewFriend.java

```
package com.lqr.wechat.model;

import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

/**
 * @ CSDN_LQR
 * @
 */
public class NewFriend {
    private NimUserInfo mUserInfo;
    private String mMsg;

    public NewFriend(NimUserInfo userInfo, String msg) {
        mUserInfo = userInfo;
        mMsg = msg;
    }

    public NimUserInfo getUserInfo() {
        return mUserInfo;
    }

    public void setUserInfo(NimUserInfo userInfo) {
        mUserInfo = userInfo;
    }

    public String getMsg() {
        return mMsg;
```

```
    }

    public void setMsg(String msg) {
        mMsg = msg;
    }
}
```

74:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\model\ResultData.java

```
package com.lqr.wechat.model;

/**
 * @ CSDN_LQR
 * @
 */
public class ResultData<T> {

    private T data;

    private int code = 200;

    private String msg;

    private Boolean success = true;

    public Boolean getSuccess() {
        return success;
    }

    public void setSuccess(Boolean success) {
        this.success = success;
    }

    public T getData() {
        return data;
    }

    public void setData(T data) {
        this.data = data;
    }

    public int getCode() {
        return code;
```

```
    }

    public void setCode(int code) {
        if (code != 200) {
            success = false;
        }
        this.code = code;
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }

}
```

75:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\model\UserCache.java

```
package com.lqr.wechat.model;

import android.content.Context;

import com.lqr.wechat.nimsdk.NimAccountSDK;
import com.netease.nimlib.sdk.StatusBarNotificationConfig;

/**
 * @ CSDN_LQR
 * @
 */
public class UserCache {

    private static Context context;
    private static String account;

    private static StatusBarNotificationConfig notificationConfig;

    public static void clear() {
        account = null;
        NimAccountSDK.removeUserInfo();
    }
```

```
    public static String getAccount() {
        return account;
    }

    public static void setAccount(String account) {
        UserCache.account = account;
    }

    public static void setNotificationConfig(StatusBarNotificationConfig notificationConfig) {
        UserCache.notificationConfig = notificationConfig;
    }

    public static StatusBarNotificationConfig getNotificationConfig() {
        return notificationConfig;
    }

    public static Context getContext() {
        return context;
    }

    public static void setContext(Context context) {
        UserCache.context = context.getApplicationContext();
    }
}
```

76:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\audio\AudioMessagePlayable.java

```
package com.lqr.wechat.nimsdk.audio;

import com.netease.nimlib.sdk.msg.attachment.AudioAttachment;
import com.netease.nimlib.sdk.msg.model.IMMessage;

public class AudioMessagePlayable implements Playable {

private IMMessage message;

public IMMessage getMessage() {
return message;
}

public AudioMessagePlayable(IMMessage playableMessage) {
```

```
this.message = playableMessage;
}

@Override
public long getDuration() {
return ((AudioAttachment) message.getAttachment()).getDuration();
}

@Override
public String getPath() {
return ((AudioAttachment) message.getAttachment()).getPath();
}

@Override
public boolean isAudioEqual(Playable audio) {
if (AudioMessagePlayable.class.isInstance(audio)) {
return message.isTheSame(((AudioMessagePlayable) audio).getMessage());
} else {
return false;
}
}
}
```

77:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\audio\BaseAudioControl.java

```
package com.lqr.wechat.nimsdk.audio;

import android.content.Context;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.Handler;
import android.text.TextUtils;

import com.lqr.wechat.R;
import com.lqr.wechat.utils.LogUtils;
import com.netease.nimlib.sdk.media.player.AudioPlayer;
import com.netease.nimlib.sdk.media.player.OnPlayListener;

abstract public class BaseAudioControl<T> {

    interface AudioControllerState {
        int stop = 0;
```

```
        int ready = 1;
        int playing = 2;
    }

    private int state;
    protected boolean isEarPhoneModeEnable = true; //

    public interface AudioControlListener {
        //AudioControlpostDelayed playRunnableAudioPlayer
        public void onAudioControllerReady(Playable playable);

        /**
         *
         */
        public void onEndPlay(Playable playable);

        /**
         *
         *
         * @param curPosition -1
         */
        public void updatePlayingProgress(Playable playable, long curPosition);
    }

    protected AudioControlListener audioControlListener;

    protected Context mContext;
    protected AudioPlayer currentAudioPlayer;
    protected Playable currentPlayable;

    protected boolean needSeek = false;
    protected long seekPosition;

    private MediaPlayer mSuffixPlayer = null;
    private boolean mSuffix = false;
    protected Handler mHandler = new Handler();

    private BasePlayerListener basePlayerListener = null;

    protected void setOnPlayListener(Playable playingPlayable, AudioControlListener
audioControlListener) {
        this.audioControlListener = audioControlListener;
```

```
    basePlayerListener = new BasePlayerListener(currentAudioPlayer, playingPlayable);
    currentAudioPlayer.setOnPlayListener(basePlayerListener);
    basePlayerListener.setAudioControlListener(audioControlListener);
}

public void setEarPhoneModeEnable(boolean isEarPhoneModeEnable) {
    this.isEarPhoneModeEnable = isEarPhoneModeEnable;
    if (isEarPhoneModeEnable) {
        updateAudioStreamType(AudioManager.STREAM_VOICE_CALL);
    } else {
        updateAudioStreamType(AudioManager.STREAM_MUSIC);
    }
}

@SuppressWarnings("unchecked")
public void changeAudioControlListener(AudioControlListener audioControlListener) {
    this.audioControlListener = audioControlListener;

    if (isPlayingAudio()) {
        OnPlayListener onPlayListener = currentAudioPlayer.getOnPlayListener();
        if (onPlayListener != null) {
            ((BasePlayerListener) onPlayListener).setAudioControlListener(audioControlListener);
        }
    }
}

public AudioControlListener getAudioControlListener() {
    return audioControlListener;
}

public BaseAudioControl(Context context, boolean suffix) {
    this.mContext = context;
    this.mSuffix = suffix;
}

protected void playSuffix() {
    if (mSuffix) {
        mSuffixPlayer = MediaPlayer.create(mContext, R.raw.audio_end_tip);
        mSuffixPlayer.setLooping(false);
        mSuffixPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        mSuffixPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
```

```
            @Override
            public void onCompletion(MediaPlayer mp) {
                mSuffixPlayer.release();
                mSuffixPlayer = null;
            }
        });
        mSuffixPlayer.start();
    }
}

protected boolean startAudio(
        Playable playable,
        AudioControlListener audioControlListener,
        int audioStreamType,
        boolean resetOrigAudioStreamType,
        long delayMillis) {
    String filePath = playable.getPath();
    if (TextUtils.isEmpty(filePath)) {
        return false;
    }

    //
    if (isPlayingAudio()) {
        stopAudio();
        //
        if (currentPlayable.isAudioEqual(playable)) {
            return false;
        }
    }

    state = AudioControllerState.stop;

    currentPlayable = playable;
    currentAudioPlayer = new AudioPlayer(mContext);
    currentAudioPlayer.setDataSource(filePath);

    setOnPlayListener(currentPlayable, audioControlListener);

    if (resetOrigAudioStreamType) {
        this.origAudioStreamType = audioStreamType;
    }
    this.currentAudioStreamType = audioStreamType;
```

```
    mHandler.postDelayed(playRunnable, delayMillis);

    state = AudioControllerState.ready;
    if (audioControlListener != null) {
        audioControlListener.onAudioControllerReady(currentPlayable);
    }

    return true;
}

Runnable playRunnable = new Runnable() {

    @Override
    public void run() {
        if (currentAudioPlayer == null) {
            LogUtils.sf("playRunnable run when currentAudioPlayer == null");
            return;
        }

        currentAudioPlayer.start(currentAudioStreamType);
    }
};

private int origAudioStreamType;
private int currentAudioStreamType;

public int getCurrentAudioStreamType() {
    return currentAudioStreamType;
}

protected int getUserSettingAudioStreamType() {
    // /
    if (isEarPhoneModeEnable) {
        return AudioManager.STREAM_VOICE_CALL;
    } else {
        return AudioManager.STREAM_MUSIC;
    }
}

protected void resetAudioController(Playable playable) {
    currentAudioPlayer.setOnPlayListener(null);
```

```
    currentAudioPlayer = null;

    state = AudioControllerState.stop;
}

//playing or ready
public boolean isPlayingAudio() {
    if (currentAudioPlayer != null) {
        return state == AudioControllerState.playing
                || state == AudioControllerState.ready;
    } else {
        return false;
    }
}

//stop or cancel
public void stopAudio() {
    if (state == AudioControllerState.playing) {
        //playing->stop
        currentAudioPlayer.stop();
    } else if (state == AudioControllerState.ready) {
        //ready->cancel
        mHandler.removeCallbacks(playRunnable);
        resetAudioController(currentPlayable);

        if (audioControlListener != null) {
            audioControlListener.onEndPlay(currentPlayable);
        }
    }
}

public boolean updateAudioStreamType(int audioStreamType) {
    if (!isPlayingAudio()) {
        return false;
    }

    if (audioStreamType == getCurrentAudioStreamType()) {
        return false;
    }

    changeAudioStreamType(audioStreamType);
    return true;
```

```
}

public boolean restoreAudioStreamType() {
    if (!isPlayingAudio()) {
        return false;
    }

    if (origAudioStreamType == getCurrentAudioStreamType()) {
        return false;
    }

    changeAudioStreamType(origAudioStreamType);
    return true;
}

private void changeAudioStreamType(int audioStreamType) {
    if (currentAudioPlayer.isPlaying()) {
        seekPosition = currentAudioPlayer.getCurrentPosition();
        needSeek = true;
        currentAudioStreamType = audioStreamType;
        currentAudioPlayer.start(audioStreamType);
    } else {
        currentAudioStreamType = origAudioStreamType;
    }
}

public class BasePlayerListener implements OnPlayListener {
    protected AudioPlayer listenerPlayingAudioPlayer;
    protected Playable listenerPlayingPlayable;
    protected AudioControlListener audioControlListener;

    public BasePlayerListener(AudioPlayer playingAudioPlayer, Playable playingPlayable) {
        listenerPlayingAudioPlayer = playingAudioPlayer;
        listenerPlayingPlayable = playingPlayable;
    }

    public void setAudioControlListener(AudioControlListener audioControlListener) {
        this.audioControlListener = audioControlListener;
    }

    protected boolean checkAudioPlayerValid() {
        if (currentAudioPlayer != listenerPlayingAudioPlayer) {
```

```
            return false;
        }

        return true;
    }

    @Override
    public void onPrepared() {
        if (!checkAudioPlayerValid()) {
            return;
        }

        state = AudioControllerState.playing;
        if (needSeek) {
            needSeek = false;
            listenerPlayingAudioPlayer.seekTo((int) seekPosition);
        }
    }

    @Override
    public void onPlaying(long curPosition) {
        if (!checkAudioPlayerValid()) {
            return;
        }

        if (audioControlListener != null) {
            audioControlListener.updatePlayingProgress(listenerPlayingPlayable, curPosition);
        }
    }

    @Override
    public void onInterrupt() {
        if (!checkAudioPlayerValid()) {
            return;
        }

        resetAudioController(listenerPlayingPlayable);
        if (audioControlListener != null) {
            audioControlListener.onEndPlay(currentPlayable);
        }

    }
```

```
    @Override
    public void onError(String error) {
        if (!checkAudioPlayerValid()) {
            return;
        }

        resetAudioController(listenerPlayingPlayable);
        if (audioControlListener != null) {
            audioControlListener.onEndPlay(currentPlayable);
        }
    }

    @Override
    public void onCompletion() {
        if (!checkAudioPlayerValid()) {
            return;
        }

        resetAudioController(listenerPlayingPlayable);
        if (audioControlListener != null) {
            audioControlListener.onEndPlay(currentPlayable);
        }

        playSuffix();
    }
}

;

public void startPlayAudio(
        T t,
        AudioControlListener audioControlListener) {
    startPlayAudio(t, audioControlListener, getUserSettingAudioStreamType());
}

public void startPlayAudio(
        T t,
        AudioControlListener audioControlListener,
        int audioStreamType) {
    startPlayAudioDelay(0, t, audioControlListener, audioStreamType);
}
```

```
    public void startPlayAudioDelay(long delayMillis, T t, AudioControlListener
audioControlListener) {
        startPlayAudioDelay(delayMillis, t, audioControlListener, getUserSettingAudioStreamType());
    }

    public abstract void startPlayAudioDelay(long delayMillis, T t, AudioControlListener
audioControlListener, int audioStreamType);

    public abstract T getPlayingAudio();
}
```

78:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\audio\MessageAudioControl.java

```
package com.lqr.wechat.nimsdk.audio;

import android.content.Context;
import android.widget.Toast;

import com.lqr.adapter.LQRAdapterForRecyclerView;
import com.lqr.wechat.nimsdk.utils.StorageUtils;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.msg.MsgService;
import com.netease.nimlib.sdk.msg.attachment.AudioAttachment;
import com.netease.nimlib.sdk.msg.constant.AttachStatusEnum;
import com.netease.nimlib.sdk.msg.constant.MsgDirectionEnum;
import com.netease.nimlib.sdk.msg.constant.MsgStatusEnum;
import com.netease.nimlib.sdk.msg.constant.MsgTypeEnum;
import com.netease.nimlib.sdk.msg.model.IMMessage;

import java.util.List;

public class MessageAudioControl extends BaseAudioControl<IMMessage> {
    private static MessageAudioControl mMessageAudioControl = null;

    private boolean mIsNeedPlayNext = false;

    private LQRAdapterForRecyclerView mAdapter = null;

    private IMMessage mItem = null;
```

```
    private MessageAudioControl(Context context) {
        super(context, true);
    }

    public static MessageAudioControl getInstance(Context context) {
        if (mMessageAudioControl == null) {
            synchronized (MessageAudioControl.class) {
                if (mMessageAudioControl == null) {
                    mMessageAudioControl = new MessageAudioControl(UIUtils.getContext());
                }
            }
        }

        return mMessageAudioControl;
    }

    @Override
    protected void setOnPlayListener(Playable playingPlayable, AudioControlListener
audioControlListener) {
        this.audioControlListener = audioControlListener;

        BasePlayerListener basePlayerListener = new BasePlayerListener(currentAudioPlayer,
playingPlayable) {

            @Override
            public void onInterrupt() {
                if (!checkAudioPlayerValid()) {
                    return;
                }

                super.onInterrupt();
                cancelPlayNext();
            }

            @Override
            public void onError(String error) {
                if (!checkAudioPlayerValid()) {
                    return;
                }

                super.onError(error);
                cancelPlayNext();
```

```
        }

        @Override
        public void onCompletion() {
            if (!checkAudioPlayerValid()) {
                return;
            }

            resetAudioController(listenerPlayingPlayable);

            boolean isLoop = false;
            if (mIsNeedPlayNext) {
                if (mAdapter != null && mItem != null) {
                    isLoop = playNextAudio(mAdapter, mItem);
                }
            }

            if (!isLoop) {
                if (audioControlListener != null) {
                    audioControlListener.onEndPlay(currentPlayable);
                }

                playSuffix();
            }
        }
    };

    basePlayerListener.setAudioControlListener(audioControlListener);
    currentAudioPlayer.setOnPlayListener(basePlayerListener);
}

@Override
public IMMessage getPlayingAudio() {
    if (isPlayingAudio() && AudioMessagePlayable.class.isInstance(currentPlayable)) {
        return ((AudioMessagePlayable) currentPlayable).getMessage();
    } else {
        return null;
    }
}

@Override
public void startPlayAudioDelay(
```

```
            long delayMillis,
            IMMessage message,
            AudioControlListener audioControlListener, int audioStreamType) {
        startPlayAudio(message, audioControlListener, audioStreamType, true, delayMillis);
    }

    //resetOrigAudioStreamType
    private void startPlayAudio(
            IMMessage message,
            AudioControlListener audioControlListener,
            int audioStreamType,
            boolean resetOrigAudioStreamType,
            long delayMillis) {
        if (StorageUtils.isExternalStorageExist()) {

            if (startAudio(new AudioMessagePlayable(message), audioControlListener,
audioStreamType, resetOrigAudioStreamType, delayMillis)) {
                // ,
                if (isUnreadAudioMessage(message)) {
                    message.setStatus(MsgStatusEnum.read);
                    NIMClient.getService(MsgService.class).updateIMMessageStatus(message);
                }
            }
        } else {
            Toast.makeText(mContext, "SD", Toast.LENGTH_SHORT).show();
        }
    }

    private boolean playNextAudio(LQRAdapterForRecyclerView tAdapter, IMMessage
messageItem) {
        List<?> list = tAdapter.getData();
        int index = 0;
        int nextIndex = -1;
        //
        for (int i = 0; i < list.size(); ++i) {
            IMMessage item = (IMMessage) list.get(i);
            if (item.equals(messageItem)) {
                index = i;
                break;
            }
        }
        //
```

```
        for (int i = index; i < list.size(); ++i) {
            IMMessage item = (IMMessage) list.get(i);
            IMMessage message = item;
            if (isUnreadAudioMessage(message)) {
                nextIndex = i;
                break;
            }
        }

        if (nextIndex == -1) {
            cancelPlayNext();
            return false;
        }
        IMMessage message = (IMMessage) list.get(nextIndex);
        AudioAttachment attach = (AudioAttachment) message.getAttachment();
        if (mMessageAudioControl != null && attach != null) {
            if (message.getAttachStatus() != AttachStatusEnum.transferred) {
                cancelPlayNext();
                return false;
            }
            if (message.getStatus() != MsgStatusEnum.read) {
                message.setStatus(MsgStatusEnum.read);
                NIMClient.getService(MsgService.class).updateIMMessageStatus(message);
            }
            //ViewHolderAudioControlListener
            //notifyDataSetChangedViewHolderViewHolderAudioControlListener
            // 1.playingAudioStreamType 2.resetOrigAudioStreamType
            mMessageAudioControl.startPlayAudio(message, null, getCurrentAudioStreamType(),
false, 0);
            mItem = (IMMessage) list.get(nextIndex);
            tAdapter.notifyDataSetChanged();
            return true;
        }
        return false;
    }

    private void cancelPlayNext() {
        setPlayNext(false, null, null);
    }

    public void setPlayNext(boolean isPlayNext, LQRAdapterForRecyclerView adapter, IMMessage
item) {
```

```
        mIsNeedPlayNext = isPlayNext;
        mAdapter = adapter;
        mItem = item;
    }

    public void stopAudio() {
        super.stopAudio();
    }

    public boolean isUnreadAudioMessage(IMMessage message) {
        if ((message.getMsgType() == MsgTypeEnum.audio)
                && message.getDirect() == MsgDirectionEnum.In
                && message.getAttachStatus() == AttachStatusEnum.transferred
                && message.getStatus() != MsgStatusEnum.read) {
            return true;
        } else {
            return false;
        }
    }
}
```

79:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\audio\Playable.java

```
package com.lqr.wechat.nimsdk.audio;

public interface Playable {
long getDuration();
String getPath();
boolean isAudioEqual(Playable audio);
}
```

80:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\custom\CustomAttachment.java

```
package com.lqr.wechat.nimsdk.custom;

import com.alibaba.fastjson.JSONObject;
import com.netease.nimlib.sdk.msg.attachment.MsgAttachment;

/**
 * @ CSDN_LQR
 * @
 */
```

```
public abstract class CustomAttachment implements MsgAttachment {

    //
    protected int type;

    CustomAttachment(int type) {
        this.type = type;
    }

    //
    public void fromJson(JSONObject data) {
        if (data != null) {
            parseData(data);
        }
    }

    //  MsgAttachment
    @Override
    public String toJson(boolean send) {
        return CustomAttachParser.packData(type, packData());
    }

    //
    protected abstract void parseData(JSONObject data);

    protected abstract JSONObject packData();
}
```

81:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\custom\CustomAttachmentType.java

```
package com.lqr.wechat.nimsdk.custom;

/**
 * @ CSDN_LQR
 * @
 */
public interface CustomAttachmentType {
    //
    int Guess = 1;//
    int SnapChat = 2;
    int Sticker = 3;
    int RTS = 4;
```

```
}

82:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\custom\CustomAttachParser.java
package com.lqr.wechat.nimsdk.custom;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.netease.nimlib.sdk.msg.attachment.MsgAttachment;
import com.netease.nimlib.sdk.msg.attachment.MsgAttachmentParser;

/**
 * @ CSDN_LQR
 * @ Application
 */
public class CustomAttachParser implements MsgAttachmentParser {

    private static final String KEY_TYPE = "type";
    private static final String KEY_DATA = "data";

    @Override
    public MsgAttachment parse(String json) {
        CustomAttachment attachment = null;
        try {
            JSONObject object = JSON.parseObject(json);
            int type = object.getInteger(KEY_TYPE);
            JSONObject data = object.getJSONObject(KEY_DATA);
            switch (type) {
                case CustomAttachmentType.Sticker:
                    attachment = new StickerAttachment();
                    break;
//                case CustomAttachmentType.Guess:
//                    attachment = new GuessAttachment();
//                    break;
//                case CustomAttachmentType.SnapChat:
//                    return new SnapChatAttachment(data);
//                case CustomAttachmentType.RTS:
//                    attachment = new RTSAttachment();
//                    break;
                default:
                    attachment = new DefaultCustomAttachment();
                    break;
```

```
            }

            if (attachment != null) {
                attachment.fromJson(data);
            }
        } catch (Exception e) {

        }

        return attachment;
    }

    public static String packData(int type, JSONObject data) {
        JSONObject object = new JSONObject();
        object.put(KEY_TYPE, type);
        if (data != null) {
            object.put(KEY_DATA, data);
        }

        return object.toJSONString();
    }
}
```

83:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\custom\DefaultCustomAttachment.java

```
package com.lqr.wechat.nimsdk.custom;

import com.alibaba.fastjson.JSONObject;

/**
 * Created by zhoujianghua on 2015/4/10.
 */
public class DefaultCustomAttachment extends CustomAttachment {

    private String content;

    public DefaultCustomAttachment() {
        super(0);
    }

    @Override
    protected void parseData(JSONObject data) {
```

```
        content = data.toJSONString();
    }

    @Override
    protected JSONObject packData() {
        JSONObject data = null;
        try {
            data = JSONObject.parseObject(content);
        } catch (Exception e) {

        }
        return data;
    }

    public String getContent() {
        return content;
    }
}
```

84:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\custom\StickerAttachment.java

```
package com.lqr.wechat.nimsdk.custom;

import com.alibaba.fastjson.JSONObject;
import com.lqr.wechat.utils.FileUtils;

/**
 * @ CSDN_LQR
 * @
 */
public class StickerAttachment extends CustomAttachment {

    private final String KEY_CATALOG = "catalog";
    private final String KEY_CHARTLET = "chartlet";

    private String catalog;
    private String chartlet;

    public StickerAttachment() {
        super(CustomAttachmentType.Sticker);
    }
```

```
    public StickerAttachment(String catalog, String emotion) {
        this();
        this.catalog = catalog;
        this.chartlet = FileUtils.getFileNameNoEx(emotion);
    }

    @Override
    protected void parseData(JSONObject data) {
        this.catalog = data.getString(KEY_CATALOG);
        this.chartlet = data.getString(KEY_CHARTLET);
    }

    @Override
    protected JSONObject packData() {
        JSONObject data = new JSONObject();
        data.put(KEY_CATALOG, catalog);
        data.put(KEY_CHARTLET, chartlet);
        return data;
    }

    public String getCatalog() {
        return catalog;
    }

    public String getChartlet() {
        return chartlet;
    }
}
```

85:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\helper\SendImageHelper.java

```
package com.lqr.wechat.nimsdk.helper;

import android.content.Context;
import android.os.AsyncTask;
import android.text.TextUtils;

import com.lqr.imagepicker.bean.ImageItem;
import com.lqr.wechat.nimsdk.utils.AttachmentStore;
import com.lqr.wechat.nimsdk.utils.ImageUtil;
import com.lqr.wechat.nimsdk.utils.StorageType;
import com.lqr.wechat.nimsdk.utils.StorageUtils;
```

```
import com.lqr.wechat.utils.FileUtils;
import com.lqr.wechat.utils.MD5Utils;
import com.lqr.wechat.utils.UIUtils;

import java.io.File;

/**
 * @ CSDN_LQR
 * @
 */
public class SendImageHelper {

    public interface Callback {
        void sendImage(File file, boolean isOrig);
    }

    public static class SendImageTask extends AsyncTask<Void, Void, File> {

        private Context mContext;
        private boolean mIsOrig;
        private ImageItem mImageItem;
        private Callback mCallback;

        public SendImageTask(Context context, boolean isOrig, ImageItem imageItem, Callback
callback) {
            mContext = context;
            mIsOrig = isOrig;
            mImageItem = imageItem;
            mCallback = callback;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected File doInBackground(Void... params) {
            String path = mImageItem.path;
            if (TextUtils.isEmpty(path)) {
                return null;
            }
```

```
    if (mIsOrig) {
        // md5
        String origMD5 = MD5Utils.decode32(path);
        String extension = FileUtils.getExtensionName(path);
        String origMD5Path = StorageUtils.getWritePath(origMD5 + "."
                + extension, StorageType.TYPE_IMAGE);
        AttachmentStore.copy(path, origMD5Path);
        //
        File imageFile = new File(origMD5Path);
        ImageUtil.makeThumbnail(mContext, imageFile);

        return new File(origMD5Path);
    } else {
        File imageFile = new File(path);
        String mimeType = FileUtils.getExtensionName(path);
        imageFile = ImageUtil.getScaledImageFileWithMD5(imageFile, mimeType);
        if (imageFile == null) {
            UIUtils.postTaskSafely(new Runnable() {
                @Override
                public void run() {
                    UIUtils.showToast("");
                }
            });
            return null;
        } else {
            ImageUtil.makeThumbnail(mContext, imageFile);
        }
        return imageFile;
    }
}

@Override
protected void onPostExecute(File file) {
    super.onPostExecute(file);

    if (file != null) {
        if (mCallback != null) {
            mCallback.sendImage(file, mIsOrig);
        }
    }
}
```

```
    }

}
```

86:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\NimAccountSDK.java

```
package com.lqr.wechat.nimsdk;

import com.lqr.wechat.AppConst;
import com.lqr.wechat.utils.SPUtils;
import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.AbortableFuture;
import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.StatusCode;
import com.netease.nimlib.sdk.auth.AuthService;
import com.netease.nimlib.sdk.auth.AuthServiceObserver;
import com.netease.nimlib.sdk.auth.LoginInfo;
import com.netease.nimlib.sdk.auth.constant.LoginSyncStatus;

/**
 * @ CSDN_LQR
 * @ SDK
 */
public class NimAccountSDK {

    private static String account;
    private static String token;

    /**
     * AbortableFuture
     */
    public static AbortableFuture<LoginInfo> login(String username, String token,
RequestCallback<LoginInfo> callback) {
        //
        LoginInfo info = new LoginInfo(username, token);
        AbortableFuture<LoginInfo> loginRequest =
NIMClient.getService(AuthService.class).login(info);
        loginRequest.setCallback(callback);
        return loginRequest;
    }
```

```
    /**
     *
     */
    public static void logout() {
        NIMClient.getService(AuthService.class).logout();
    }

    /**
     *
     */
    public static void onlineStatusListen(Observer<StatusCode> observer, boolean register) {
        NIMClient.getService(AuthServiceObserver.class).observeOnlineStatus(
                observer, register);
    }

    /**
     *
     * <p>
     * SDK
     *
     * @param register
     */
    public static void syncDataListen(Observer<LoginSyncStatus> observer, boolean register) {
        NIMClient.getService(AuthServiceObserver.class).observeLoginSyncDataStatus(observer,
register);
    }

    public static String getUserAccount() {
        account =
SPUtils.getInstance(UIUtils.getContext()).getString(AppConst.Account.KEY_USER_ACCOUNT,
"");
        return account;
    }

    public static String getUserToken() {
        token =
SPUtils.getInstance(UIUtils.getContext()).getString(AppConst.Account.KEY_USER_TOKEN, "");
        return token;
    }

    public static void saveUserAccount(String account) {
```

```
        NimAccountSDK.account = account;
SPUtils.getInstance(UIUtils.getContext()).putString(AppConst.Account.KEY_USER_ACCOUNT, a
ccount);
    }

    public static void saveUserToken(String token) {
        NimAccountSDK.token = token;
        SPUtils.getInstance(UIUtils.getContext()).putString(AppConst.Account.KEY_USER_TOKEN,
token);
    }

    public static void removeUserInfo() {
SPUtils.getInstance(UIUtils.getContext()).remove(AppConst.Account.KEY_USER_ACCOUNT);
  SPUtils.getInstance(UIUtils.getContext()).remove(AppConst.Account.KEY_USER_TOKEN);
    }
}
```

87:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\NimBlackListSDK.java

```
package com.lqr.wechat.nimsdk;

import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.friend.FriendService;
import com.netease.nimlib.sdk.friend.FriendServiceObserve;
import com.netease.nimlib.sdk.friend.model.BlackListChangedNotify;

import java.util.List;

/**
 * @ CSDN_LQR
 * @ SDK
 */
public class NimBlackListSDK {

    /**
     *
     */
    public static void addToBlackList(String account, RequestCallback<Void> callback) {
        NIMClient.getService(FriendService.class).addToBlackList(account)
                .setCallback(callback);
```

```
    }

    /**
     *
     */
    public static void removeFromBlackList(String account, RequestCallback<Void> callback) {
        NIMClient.getService(FriendService.class).removeFromBlackList(account)
                .setCallback(callback);
    }

    /**
     *
     */
    public static List<String> getBlackList() {
        return NIMClient.getService(FriendService.class).getBlackList();
    }

    /**
     *
     */
    public static boolean isInBlackList(String account) {
        boolean black = NIMClient.getService(FriendService.class).isInBlackList(account);
        return black;
    }

    /**
     * /
     */
    public static void observeBlackListChangedNotify(Observer<BlackListChangedNotify>
blackListChangedNotifyObserver, boolean register) {
        NIMClient.getService(FriendServiceObserve.class)
                .observeBlackListChangedNotify(blackListChangedNotifyObserver, register);
    }

}
```

88:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\NimFriendSDK.java

```
package com.lqr.wechat.nimsdk;

import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.Observer;
```

```
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.friend.FriendService;
import com.netease.nimlib.sdk.friend.FriendServiceObserve;
import com.netease.nimlib.sdk.friend.constant.FriendFieldEnum;
import com.netease.nimlib.sdk.friend.constant.VerifyType;
import com.netease.nimlib.sdk.friend.model.AddFriendData;
import com.netease.nimlib.sdk.friend.model.Friend;
import com.netease.nimlib.sdk.friend.model.FriendChangedNotify;

import java.util.List;
import java.util.Map;

/**
 * @ CSDN_LQR
 * @ SDK
 */
public class NimFriendSDK {


    /**
     *
     * <p>
     *  VerifyType AddFriendData
     *
     */
    public static void addFriend(String account, String msg, RequestCallback<Void> callback) {
        final VerifyType verifyType = VerifyType.VERIFY_REQUEST; //
        NIMClient.getService(FriendService.class).addFriend(new AddFriendData(account,
verifyType, msg))
                .setCallback(callback);
    }

    /**
     * /
     * <p>
     *  AddFriendNotify.Event.RECV_AGREE_ADD_FRIEND
AddFriendNotify.Event.RECV_REJECT_ADD_FRIEND
     */
    public static void ackAddFriendRequest(String account, boolean agree) {
        NIMClient.getService(FriendService.class).ackAddFriendRequest(account, true); //
    }
```

```
    /**
     *
     * <p>
     * ,
     */
    public static List<String> getFriendAccounts() {
        List<String> friendAccounts =
NIMClient.getService(FriendService.class).getFriendAccounts();
        return friendAccounts;
    }

    /**
     *
     */
    public static List<Friend> getFriends() {
        List<Friend> friends = NIMClient.getService(FriendService.class).getFriends();
        return friends;
    }

    /**
     *
     * <p>
     *
     */
    public static void deleteFriend(String account, RequestCallback<Void> callback) {
        NIMClient.getService(FriendService.class).deleteFriend(account)
                .setCallback(callback);
    }

    /**
     *
     */
    public static Friend getFriendByAccount(String account) {
        Friend friend = NIMClient.getService(FriendService.class).getFriendByAccount(account);
        return friend;
    }

    /**
     *
     */
    public static boolean isMyFriend(String account) {
        boolean isMyFriend = NIMClient.getService(FriendService.class).isMyFriend(account);
```

```
        return isMyFriend;
    }

    /**
     *
     *  FriendFieldEnum
     */
    public static void updateFriendFields(String account, Map<FriendFieldEnum, Object> map,
RequestCallback<Void> callback) {
        NIMClient.getService(FriendService.class).updateFriendFields(account, map)
                .setCallback(callback);
    }

    /**
     *
     * <p>
     *  APP  APP
     */
    public static void observeFriendChangedNotify(Observer<FriendChangedNotify>
friendChangedNotifyObserver, boolean register) {
NIMClient.getService(FriendServiceObserve.class).observeFriendChangedNotify(friendChangedN
otifyObserver, register);
    }


}
```

89:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\NimHistorySDK.java

```
package com.lqr.wechat.nimsdk;

import com.netease.nimlib.sdk.InvocationFuture;
import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.msg.MsgService;
import com.netease.nimlib.sdk.msg.constant.MsgTypeEnum;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.msg.model.IMMessage;
import com.netease.nimlib.sdk.msg.model.QueryDirectionEnum;

import java.util.List;
```

```
import static com.netease.nimlib.sdk.NIMClient.getService;

/**
 * @ CSDN_LQR
 * @
 */
public class NimHistorySDK {

    /**
     * SDK  (direct) anchor  (QUERY_OLD)  (QUERY_NEW) anchor  limit  asc  time
     *  MessageBuilder#createEmptyMessage
     *
     * @param anchor    IMMessage
     * @param direction QueryDirectionEnum
     * @param limit     int
     * @param asc       boolean  true false
     * @return
     */
    public static InvocationFuture<List<IMMessage>> queryMessageListEx(IMMessage anchor,
QueryDirectionEnum direction, int limit, boolean asc) {
        return getService(MsgService.class).queryMessageListEx(anchor, direction, limit, asc);
    }

    /**
     * <br>
     *  (direction) anchor  anchor  toTime  anchor  limit <br>
     *  toTime  limit  toTime  limit  limit  limit <br>
     * direction  QUERY_OLD  direction  QUERY_NEW <br>
     *
     *
     * @param anchor
     * @param toTime     QUERY_OLDtoTime  anchor.getTime() QUERY_NEWtoTime
anchor.getTime() <br>
     * @param direction
     * @param limit
     * @return
     */
    public static InvocationFuture<List<IMMessage>> queryMessageListExTime(IMMessage
anchor, long toTime, QueryDirectionEnum direction, int limit) {
        return getService(MsgService.class).queryMessageListExTime(anchor, toTime, direction,
limit);
    }
```

```
    /**
     * uuidIMMessage()
     */
    public static InvocationFuture<List<IMMessage>> queryMessageListByUuidAsync(List<String>
uuids) {
        return getService(MsgService.class).queryMessageListByUuid(uuids);
    }

    /**
     * uuidIMMessage()
     */
    public static List<IMMessage> queryMessageListByUuidSync(List<String> uuids) {
        return getService(MsgService.class).queryMessageListByUuidBlock(uuids);
    }

    /**
     *  msgTypeEnum  anchor  sessionId  anchor  limit
     *
     * @param msgTypeEnum MsgTypeEnum
     * @param anchor      IMMessage
     * @param limit       int
     * @return
     */
    public static InvocationFuture<List<IMMessage>> queryMessageListByType(MsgTypeEnum
msgTypeEnum, IMMessage anchor, int limit) {
        return NIMClient.getService(MsgService.class).queryMessageListByType(msgTypeEnum,
anchor, limit);
    }

    /**
     *
     *
     * @param keyword      String
     * @param fromAccounts List<String>
     *                     keyword
     * @param anchor       IMMessage
     * @param limit        int
     * @return
     */
    public static void searchMessageHistory(String keyword, List<String> fromAccounts,
IMMessage anchor, int limit, RequestCallbackWrapper<List<IMMessage>> callback) {
```

```
        NIMClient.getService(MsgService.class).searchMessageHistory(keyword, fromAccounts,
anchor, limit)
                .setCallback(callback);
    }

    /**
     *
     *
     * @param keyword
     * @param fromAccounts keyword
     * @param time         time
     * @param limit
     * @return InvocationFuture
     */
    public static void searchAllMessageHistory(String keyword, List<String> fromAccounts, long
time, int limit, RequestCallbackWrapper<List<IMMessage>> callback) {
        NIMClient.getService(MsgService.class).searchAllMessageHistory(keyword, fromAccounts,
time, limit)
                .setCallback(callback);
    }

    /**
     *
     */
    public static void deleteChattingHistory(IMMessage message) {
        NIMClient.getService(MsgService.class).deleteChattingHistory(message);
    }

    /**
     *
     */
    public static void clearChattingHistory(String account, SessionTypeEnum sessionType) {
        NIMClient.getService(MsgService.class).clearChattingHistory(account, sessionType);
    }

    /**
     *
     *
     * @param anchor    IMMessage  null
     * @param toTime    long
     * @param limit     int ( 100 )
     * @param direction QueryDirectionEnum
```

```
     *                QUERY_OLD
     *                QUERY_NEW
     * @param persist   boolean
     * @return InvocationFuture
     */
    public static InvocationFuture<List<IMMessage>> pullMessageHistoryEx(IMMessage anchor,
long toTime, int limit, QueryDirectionEnum direction, boolean persist) {
        return NIMClient.getService(MsgService.class).pullMessageHistoryEx(anchor, toTime, limit,
direction, persist);
    }

    /**
     *  anchor
     *  limit
     *
     * @param anchor  IMMessage
     * @param limit   int ( 100 )
     * @param persist boolean
     * @return InvocationFuture
     */
    public static InvocationFuture<List<IMMessage>> pullMessageHistory(IMMessage anchor, int
limit, boolean persist) {
        return NIMClient.getService(MsgService.class).pullMessageHistory(anchor, limit, persist);
    }

}
```

90:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\NimMessageSDK.java

```
package com.lqr.wechat.nimsdk;

import android.media.MediaPlayer;
import android.net.Uri;
import android.text.TextUtils;
import android.widget.Toast;

import com.lqr.wechat.utils.UIUtils;
import com.netease.nimlib.sdk.AbortableFuture;
import com.netease.nimlib.sdk.InvocationFuture;
import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
```

```
import com.netease.nimlib.sdk.msg.MessageBuilder;
import com.netease.nimlib.sdk.msg.MsgService;
import com.netease.nimlib.sdk.msg.MsgServiceObserve;
import com.netease.nimlib.sdk.msg.attachment.FileAttachment;
import com.netease.nimlib.sdk.msg.attachment.MsgAttachment;
import com.netease.nimlib.sdk.msg.constant.AttachStatusEnum;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.msg.model.IMMessage;

import java.io.File;
import java.util.List;
import java.util.Map;

/**
 * @ CSDN_LQR
 * @ SDK
 */
public class NimMessageSDK {

    /**
     *
     *
     * @param sessionId    ID ID
     * @param sessionType
     * @param content
     * @return
     */
    public static IMMessage createTextMessage(String sessionId, SessionTypeEnum sessionType,
String content) {
        return MessageBuilder.createTextMessage(
                sessionId,
                sessionType,
                content
        );
    }

    /**
     *
     *
     * @param sessionId    ID ID
     * @param sessionType
     * @param latitude
```

```
     * @param longitude
     * @param address
     * @return
     */
    public static IMMessage createLocationMessage(String sessionId, SessionTypeEnum
sessionType, double latitude, double longitude, String address) {
        return MessageBuilder.createLocationMessage(
                sessionId,
                sessionType,
                latitude,
                longitude,
                address
        );
    }

    /**
     *
     *
     * @param sessionId    ID ID
     * @param sessionType
     * @param file
     * @param displayName  APP  null
     * @return
     */
    public static IMMessage createImageMessage(String sessionId, SessionTypeEnum
sessionType, File file, String displayName) {
        return MessageBuilder.createImageMessage(
                sessionId,
                sessionType,
                file,
                displayName
        );
    }

    /**
     *
     *
     * @param sessionId    ID ID
     * @param sessionType
     * @param file
     * @param duration    ms
     * @return
```

```
     */
    public static IMMessage createAudioMessage(String sessionId, SessionTypeEnum
sessionType, File file, long duration) {
        return MessageBuilder.createAudioMessage(
                sessionId,
                sessionType,
                file,
                duration
        );
    }

    /**
     * ()
     *
     * @param sessionId    ID ID
     * @param sessionType
     * @param file
     * @return
     */
    public static IMMessage createAudioMessage(String sessionId, SessionTypeEnum
sessionType, File file) {
        MediaPlayer mediaPlayer = getVideoMediaPlayer(file);
        long duration = mediaPlayer == null ? 0 : mediaPlayer.getDuration();
        return createAudioMessage(sessionId, sessionType, file, duration);
    }

    /**
     *
     *
     * @param sessionId    ID ID
     * @param sessionType
     * @param file
     * @param displayName
     * @return
     */
    public static IMMessage createVideoMessage(String sessionId, SessionTypeEnum
sessionType, File file, String displayName) {
        MediaPlayer mediaPlayer = getVideoMediaPlayer(file);
        long duration = mediaPlayer == null ? 0 : mediaPlayer.getDuration();
        int height = mediaPlayer == null ? 0 : mediaPlayer.getVideoHeight();
        int width = mediaPlayer == null ? 0 : mediaPlayer.getVideoWidth();
        return MessageBuilder.createVideoMessage(
```

```
                sessionId,
                sessionType,
                file,
                duration, //
                width, //
                height, //
                displayName
        );
    }

    /**
     * //
     * // Tip
     * // setAttachmentAttachment
     *
     * @param sessionId    ID ID
     * @param sessionType
     * @param content
     * @return
     */
    public static IMMessage createTipMessage(String sessionId, SessionTypeEnum sessionType,
String content) {
        IMMessage message = MessageBuilder.createTipMessage(
                sessionId,
                sessionType
        );
        if (!TextUtils.isEmpty(content))
            message.setContent(content);
        return message;
    }

    /**
     *
     *
     * @param sessionId    ID ID
     * @param sessionType
     * @param content
     * @param attachment
     * @return
     */
    public static IMMessage createCustomMessage(String sessionId, SessionTypeEnum
sessionType, String content, MsgAttachment attachment) {
```

```
    return MessageBuilder.createCustomMessage(sessionId, sessionType, content, attachment);
}

/**
 *
 *
 * @param message
 * @param data
 * @return
 */
public static IMMessage setRemoteExtension(IMMessage message, Map data) {
    message.setRemoteExtension(data);
    return message;
}

/**
 *
 *
 * @param message
 * @param data
 * @return
 */
public static IMMessage setLocalExtension(IMMessage message, Map data) {
    message.setLocalExtension(data);
    return message;
}

/**
 *
 *
 * @param message
 * @param data
 * @return
 */
public static IMMessage setPushPayload(IMMessage message, Map data) {
    message.setPushPayload(data);
    return message;
}

/**
 *
 *
```

```
     * @param message
     * @param pushContent
     * @return
     */
    public static IMMessage setPushContent(IMMessage message, String pushContent) {
        message.setPushContent(pushContent);
        return message;
    }

    /**
     *
     *
     * @param message
     * @param resend  truefalse
     */
    private static void sendMessage(IMMessage message, boolean resend) {
        NIMClient.getService(MsgService.class).sendMessage(message, resend);
    }

    /**
     *
     *
     * @param message
     */
    public static void sendMessage(IMMessage message) {
        sendMessage(message, false);
    }

    /**
     *
     *
     * @param message
     */
    public static void reSendMessage(IMMessage message) {
        sendMessage(message, true);
    }

    /**
     *
     *
     * @param forwardMessage
     * @param sessionId       id
```

```
     * @param sessionTypeEnum
     */
    public static void forwardMessage(IMMessage forwardMessage, String sessionId,
SessionTypeEnum sessionTypeEnum) {
        //
        IMMessage message = MessageBuilder.createForwardMessage(forwardMessage,
sessionId, sessionTypeEnum);
        if (message == null) {
            Toast.makeText(UIUtils.getContext(), "", Toast.LENGTH_SHORT).show();
            return;
        }
        sendMessage(message, false);
    }

    /**
     *
     * 1. APP MsgService#saveMessageToLocal UI notify  true  #observeReceiveMessage
     * 2.  1.8.0  IMMessage  CustomMessageConfig  enableUnreadCount  false
     *
     * @param message
     * @param nofity
     */
    public static void saveMessageToLocal(IMMessage message, boolean nofity) {
        NIMClient.getService(MsgService.class).saveMessageToLocal(message, nofity);
    }

    /**
     * /
     * <p>
     *
     * // 1sessionId
     * // 2
     * // 3
     *
     * @param observer  msgStatus  attachStatus
     * @param register truefalse
     */
    public static void observeMsgStatus(Observer<IMMessage> observer, boolean register) {
        NIMClient.getService(MsgServiceObserve.class).observeMsgStatus(observer, register);
    }

    /**
```

```
     *
     *
     * @param observer  progress  uuid  UI)
     */
    public static void observeAttachProgress(Observer observer, boolean register) {
        NIMClient.getService(MsgServiceObserve.class).observeAttachmentProgress(observer,
register);
    }

    /**
     *
     *  APP
     * <p>
     *  onCreate  onDestroy
     *
     * @param incomingMessageObserver
     */
    public static void observeReceiveMessage(Observer<List<IMMessage>>
incomingMessageObserver, boolean register) {
        NIMClient.getService(MsgServiceObserve.class)
                .observeReceiveMessage(incomingMessageObserver, register);
    }

    /**
     *
     * <p>
     * 414
     * 414
     *
     * @param message
     * @return
     */
    public static boolean isOriginImageHasDownloaded(final IMMessage message) {
        if (message.getAttachStatus() == AttachStatusEnum.transferred) {
            if (message.getAttachment() instanceof FileAttachment) {
                if (!TextUtils.isEmpty(((FileAttachment) message.getAttachment()).getPath())) {
                    return true;
                }
            }
        }
        return false;
    }
```

```
    /**
     *
     *
     * @param message
     * @param thumb   true<br>
     *
     * @return AbortableFuture
     */
    public static AbortableFuture downloadAttachment(IMMessage message, boolean thumb) {
        return NIMClient.getService(MsgService.class).downloadAttachment(message, true);
    }

    /**
     *  <br>
     * <br>
     * {@link MsgServiceObserve#observeRecentContactDeleted(Observer, boolean)}
     *
     * @param clearRecent true
     */
    public static void clearMsgDatabase(boolean clearRecent) {
        NIMClient.getService(MsgService.class).clearMsgDatabase(clearRecent);
    }

    /**
     *
     *
     * @param message
     * @return InvocationFuture
     */
    public static InvocationFuture<Void> revokeMessage(IMMessage message,
RequestCallback<Void> callback) {
        InvocationFuture<Void> voidInvocationFuture = NIMClient.getService(MsgService.class)
                .revokeMessage(message);
        voidInvocationFuture.setCallback(callback);
        return voidInvocationFuture;
    }

    /**
     *
     *
     * @param message
```

```
     * @param sessionId   ID
     * @param sessionType   @return
     */
    public static boolean isCurrentSessionMessage(IMMessage message, String sessionId,
SessionTypeEnum sessionType) {
        return message.getSessionType() == sessionType
                && message.getSessionId() != null
                && message.getSessionId().equals(sessionId);
    }

    /**
     * mediaPlayer
     *
     * @param file
     * @return mediaPlayer
     */
    private static MediaPlayer getVideoMediaPlayer(File file) {
        try {
            return MediaPlayer.create(UIUtils.getContext(), Uri.parse("file://" + file.getAbsolutePath()));
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }


}
```

91:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\NimRecentContactSDK.java

```
package com.lqr.wechat.nimsdk;

import com.netease.nimlib.sdk.InvocationFuture;
import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.msg.MsgService;
import com.netease.nimlib.sdk.msg.MsgServiceObserve;
import com.netease.nimlib.sdk.msg.constant.SessionTypeEnum;
import com.netease.nimlib.sdk.msg.model.RecentContact;
```

```
import java.util.List;

/**
 * @ CSDN_LQR
 * @ SDK
 * <p>
 *  RecentContact
 *
 * RecentContact  tag extension Map@
 *  SDK  SDK
 *
 *  SDK   MsgService#saveMessageToLocal
 */
public class NimRecentContactSDK {

    /**
     *
     *
     * @param callback
     */
    public static void queryRecentContacts(RequestCallbackWrapper<List<RecentContact>>
callback) {
        NIMClient.getService(MsgService.class).queryRecentContacts()
                .setCallback(callback);
    }

    /**
     * /
     *
     * @param messageObserver
     * @param register        /
     */
    public static void observeRecentContact(Observer<List<RecentContact>> messageObserver,
boolean register) {
        //  /
        NIMClient.getService(MsgServiceObserve.class)
                .observeRecentContact(messageObserver, register);
    }

    /**
     *
     *
```

```
     * @return
     */
    public static int getTotalUnreadCount() {
        int unreadNum = NIMClient.getService(MsgService.class).getTotalUnreadCount();
        return unreadNum;
    }

    /**
     * ()<br>
     * {@link MsgServiceObserve#observeRecentContact(Observer, boolean)}
     *
     * @param account
     * @param sessionType
     */
    public static void clearUnreadCount(String account, SessionTypeEnum sessionType) {
        NIMClient.getService(MsgService.class).clearUnreadCount(account, sessionType);
    }

    /**
     *
     * <p>
     *  setChattingAccount SDK
     *
     * @param sessionId
     * @param sessionType
     */
    public static void setChattingAccount(String sessionId, SessionTypeEnum sessionType) {
        NIMClient.getService(MsgService.class).setChattingAccount(sessionId, sessionType);
    }

    /**
     *
     *
     * @param recent
     */
    public static void deleteRecentContact(RecentContact recent) {
        NIMClient.getService(MsgService.class).deleteRecentContact(recent);
    }

    /**
     *  MsgServiceObserve#observeRecentContactDeleted
     *
```

```
     * @param account
     * @param sessionType
     */
    public static void deleteRecentContactAndNotify(String account, SessionTypeEnum
sessionType) {
        NIMClient.getService(MsgService.class).deleteRecentContact2(account, sessionType);
    }

    /**
     *
     *
     *
     * @param contactId       IDID
     * @param sessionTypeEnum
     * @return InvocationFuture
     */
    public static InvocationFuture<Void> deleteRoamingRecentContact(String contactId,
SessionTypeEnum sessionTypeEnum, RequestCallback<Void> callback) {
        InvocationFuture<Void> voidInvocationFuture = NIMClient.getService(MsgService.class)
                .deleteRoamingRecentContact(contactId, sessionTypeEnum);
        voidInvocationFuture.setCallback(callback);
        return voidInvocationFuture;
    }
}
```

92:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\NimSystemSDK.java

```
package com.lqr.wechat.nimsdk;

import com.netease.nimlib.sdk.InvocationFuture;
import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.msg.SystemMessageObserver;
import com.netease.nimlib.sdk.msg.SystemMessageService;
import com.netease.nimlib.sdk.msg.constant.SystemMessageType;
import com.netease.nimlib.sdk.msg.model.SystemMessage;

import java.util.List;

/**
 * @ CSDN_LQR
 * @ SDK
```

```
 */
public class NimSystemSDK {


    /**
     *
     */
    public static void observeReceiveSystemMsg(Observer<SystemMessage>
systemMessageObserver, boolean register) {
NIMClient.getService(SystemMessageObserver.class).observeReceiveSystemMsg(systemMessa
geObserver, register);
    }

    /**
     *
     */
    public static List<SystemMessage> querySystemMessagesBlock(int offset, int limit) {
        return NIMClient.getService(SystemMessageService.class)
                .querySystemMessagesBlock(offset, limit);// offsetoffsetlimitlimit
    }

    /**
     *
     */
    public static InvocationFuture<List<SystemMessage>>
querySystemMessageByType(List<SystemMessageType> types, int offset, int limit) {
        return
NIMClient.getService(SystemMessageService.class).querySystemMessageByType(types, offset,
limit);
    }

    /**
     *
     */
    public static void deleteSystemMessage(SystemMessage message) {
        NIMClient.getService(SystemMessageService.class)
                .deleteSystemMessage(message.getMessageId());
    }

    /**
     *
     */
```

```
public static void clearSystemMessages() {
    NIMClient.getService(SystemMessageService.class).clearSystemMessages();
}

/**
 *
 * <p>
 * “”
 */
public static void clearSystemMessagesByType(List<SystemMessageType> types) {
    NIMClient.getService(SystemMessageService.class).clearSystemMessagesByType(types);
}

/**
 *
 */
public int querySystemMessageUnreadCountBlock() {
    int unread = NIMClient.getService(SystemMessageService.class)
            .querySystemMessageUnreadCountBlock();
    return unread;
}

/**
 *
 * <p>
 * “”
 */
public static int querySystemMessageUnreadCountByType(List<SystemMessageType> types) {
    int unread = NIMClient.getService(SystemMessageService.class)
            .querySystemMessageUnreadCountByType(types);
    return unread;
}

/**
 *
 */
public static void setSystemMessageRead(long messageId) {
    NIMClient.getService(SystemMessageService.class).setSystemMessageRead(messageId);
}

/**
 *
```

```
     * <p>
     * 0
     */
    public static void resetSystemMessageUnreadCount() {
        NIMClient.getService(SystemMessageService.class).resetSystemMessageUnreadCount();
    }

    /**
     *
     * <p>
     * “”
     */
    public static void resetSystemMessageUnreadCount(List<SystemMessageType> types) {
NIMClient.getService(SystemMessageService.class).resetSystemMessageUnreadCountByType(t
ypes);
    }
}
```

93:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\NimTeamSDK.java

```
package com.lqr.wechat.nimsdk;

import android.text.TextUtils;

import com.lqr.wechat.AppConst;
import com.lqr.wechat.model.UserCache;
import com.netease.nimlib.sdk.InvocationFuture;
import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.friend.model.Friend;
import com.netease.nimlib.sdk.msg.model.SystemMessage;
import com.netease.nimlib.sdk.team.TeamService;
import com.netease.nimlib.sdk.team.TeamServiceObserver;
import com.netease.nimlib.sdk.team.constant.TeamAllMuteModeEnum;
import com.netease.nimlib.sdk.team.constant.TeamFieldEnum;
import com.netease.nimlib.sdk.team.constant.TeamTypeEnum;
import com.netease.nimlib.sdk.team.constant.VerifyTypeEnum;
import com.netease.nimlib.sdk.team.model.MemberChangeAttachment;
import com.netease.nimlib.sdk.team.model.MuteMemberAttachment;
import com.netease.nimlib.sdk.team.model.Team;
```

```
import com.netease.nimlib.sdk.team.model.TeamMember;
import com.netease.nimlib.sdk.team.model.UpdateTeamAttachment;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.io.Serializable;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * @ CSDN_LQR
 * @ SDK
 * <p>
 * 1
 * <p>
 *  Demo
 * <p>
 * 2
 * <p>
 * 2.4.02.4.0
 */
public class NimTeamSDK {

    /**
     *
     * <p>
     *
     *  iOS  APNS
     *  Team  mute
     *
     */
    public static void muteTeam(String teamId, boolean mute) {
        NIMClient.getService(TeamService.class).muteTeam(teamId, mute);
    }

    /**
     * ()
     * <p>
     *
     */
    public static void queryTeamList(RequestCallbackWrapper<List<Team>> callback) {
        NIMClient.getService(TeamService.class).queryTeamList()
```

```
                .setCallback(callback);
    }

    /**
     * ()
     * <p>
     */
    public static List<Team> queryTeamListBlock() {
        List<Team> teams = NIMClient.getService(TeamService.class).queryTeamListBlock();
        return teams;
    }

    /**
     *
     */
    public static void queryTeamListByType(TeamTypeEnum type, RequestCallback<List<Team>>
callback) {
        NIMClient.getService(TeamService.class).queryTeamListByType(type)
                .setCallback(callback);
    }

    /**
     * ID()
     * <p>
     *  searchTeam
     */
    public static void queryTeam(String teamId, RequestCallbackWrapper<Team> callback) {
        NIMClient.getService(TeamService.class).queryTeam(teamId).setCallback(callback);
    }

    /**
     * ID()
     * <p>
     *  searchTeam
     */
    public static Team queryTeamBlock(String teamId) {
        Team team = NIMClient.getService(TeamService.class).queryTeamBlock(teamId);
        return team;
    }

    /**
     *
```

```
     * <p>
     *
     * <p>
     * 1024414
     *
     * @param fields   fields key value  field  fieldType
     * @param type     TeamTypeEnum.Advanced,TeamTypeEnum.Normal
     * @param accounts
     * @param callback
     */
    public static void createTeam(HashMap<TeamFieldEnum, Serializable> fields, TeamTypeEnum
type, List<String> accounts, RequestCallback<Team> callback) {
        NIMClient.getService(TeamService.class).createTeam(fields, type, "", accounts)
                .setCallback(callback);
    }

    /**
     *
     */
    public static void applyJoinTeam(String teamId, String reason, RequestCallback<Team>
callback) {
        NIMClient.getService(TeamService.class)
                .applyJoinTeam(teamId, reason)
                .setCallback(callback);
    }

    /**
     *
     * <p>
     *
     */
    public static void dismissTeam(String teamId, RequestCallback<Void> callback) {
        NIMClient.getService(TeamService.class).dismissTeam(teamId)
                .setCallback(callback);
    }

    /**
     *
     * <p>
     * SDK 2.4.0 SDK 2.4.0
     * <p>
     * onSuccess
```

```
     * <p>
     * onFailed810 onSuccess ()
     */
    public static void addMembers(String teamId, List<String> accounts, RequestCallback<Void>
callback) {
        NIMClient.getService(TeamService.class).addMembers(teamId, accounts)
                .setCallback(callback);
    }

    /**
     * ()
     * <p>
     *
     * <p>
     * () notification  IMMessage NotificationType#KickMember,  MemberChangeAttachment
     * MemberChangeAttachment#getExtension
     */
    public static void removeMember(String teamId, String account, RequestCallback<Void>
callback) {
        NIMClient.getService(TeamService.class).removeMember(teamId, account)
                .setCallback(callback);
    }

    /**
     *
     */
    public static InvocationFuture<Void> removeMembers(String teamId, List<String> members) {
        InvocationFuture<Void> invocationFuture =
NIMClient.getService(TeamService.class).removeMembers(teamId, members);
        return invocationFuture;
    }

    /**
     *
     * <p>
     *
     * <p>
     * () notification  IMMessage MemberChangeAttachment
     */
    public static void quitTeam(String teamId, RequestCallback<Void> callback) {
        NIMClient.getService(TeamService.class).quitTeam(teamId)
                .setCallback(callback);
```

```
    }

    /**
     *
     * <p>
     *
     *
     * @param teamId   ID
     * @param account
     * @param quit
     * @param callback
     */
    public static void transferTeam(String teamId, String account, boolean quit,
RequestCallback<List<TeamMember>> callback) {
        NIMClient.getService(TeamService.class)
                .transferTeam(teamId, account, quit)
                .setCallback(callback);
    }

    /**
     *
     * <p>
     *
     * <p>
     *
     */
    public static void acceptInvite(SystemMessage message, RequestCallback<Void> callback) {
        NIMClient.getService(TeamService.class)
                .acceptInvite(message.getTargetId(), message.getFromAccount())
                .setCallback(callback);
    }

    /**
     * ,
     *  SystemMessageType#DeclineTeamInvite
     * <p>
     *
     * <p>
     *
     */
    public static void declineInvite(SystemMessage message, RequestCallback<Void> callback,
String reason) {
```

```
        NIMClient.getService(TeamService.class)
                .declineInvite(message.getTargetId(), message.getFromAccount(), reason)
                .setCallback(callback);
    }

    /**
     *
     * () notification  IMMessage MemberChangeAttachment
     * <p>
     *
     * <p>
     *  SystemMessageType#TeamApply
     */
    public static void passApply(SystemMessage message, RequestCallback<Void> callback) {
        NIMClient.getService(TeamService.class)
                .passApply(message.getTargetId(), message.getFromAccount())
                .setCallback(callback);
    }

    /**
     *
     *  SystemMessageType#RejectTeamApply
     * <p>
     *
     * <p>
     *  SystemMessageType#TeamApply
     */
    public static void rejectApply(SystemMessage message, String reason, RequestCallback<Void>
callback) {
        NIMClient.getService(TeamService.class)
                .rejectApply(message.getTargetId(), message.getFromAccount(), reason)
                .setCallback(callback);
    }

    /**
     *
     */
    public static void updateTeamField(String teamId, TeamFieldEnum teamFieldEnum,
Serializable value, RequestCallback<Void> callback) {
        NIMClient.getService(TeamService.class).updateTeam(teamId, teamFieldEnum, value)
                .setCallback(callback);
    }
```

```
    /**
     *
     */
    public static InvocationFuture<Void> updateTeamFields(String teamId, Map<TeamFieldEnum,
Serializable> fields) {
        InvocationFuture<Void> voidInvocationFuture =
NIMClient.getService(TeamService.class).updateTeamFields(teamId, fields);
        return voidInvocationFuture;
    }

    /**
     *
     * <p>
     *
     *
     *
     *
     * @param teamId  ID
     * @param account
     * @param nick
     * @return InvocationFuture
     */
    public static void updateMemberNick(String teamId, String account, String nick,
RequestCallback<Void> callback) {
        NIMClient.getService(TeamService.class).updateMemberNick(teamId, account,
nick).setCallback(callback);
    }

    /**
     *
     * <p>
     *
     *
     * @param teamId ID
     * @param nick
     * @return InvocationFuture
     */
    public static void updateMyTeamNick(String teamId, String nick, RequestCallback<Void>
callback) {
        NIMClient.getService(TeamService.class).updateMyTeamNick(teamId,
nick).setCallback(callback);
```

```
    }

    /**
     *
     * <p>
     *
     *
     * @param teamId ID
     * @param extMap Map<String,Object>
     * @return InvocationFuture
     */
    public static void updateMyMemberExtension(String teamId, Map<String, Object> extMap,
RequestCallback<Void> callback) {
        NIMClient.getService(TeamService.class).updateMyMemberExtension(teamId,
extMap).setCallback(callback);
    }

    /**
     * /
     */
    public static void observeTeamUpdate(Observer<List<Team>> teamUpdateObserver, boolean
register) {
NIMClient.getService(TeamServiceObserver.class).observeTeamUpdate(teamUpdateObserver, re
gister);
    }

    /**
     * /
     */
    public static void observeTeamRemove(Observer<Team> teamRemoveObserver, boolean
register) {
NIMClient.getService(TeamServiceObserver.class).observeTeamRemove(teamRemoveObserver,
register);
    }

    /**
     * /
     */
    public static void observeMemberUpdate(Observer<List<TeamMember>>
memberUpdateObserver, boolean register) {
NIMClient.getService(TeamServiceObserver.class).observeMemberUpdate(memberUpdateObser
ver, register);
```

```
    }

    /**
     * /
     */
    public static void observeMemberRemove(Observer<TeamMember> memberRemoveObserver,
boolean register) {
NIMClient.getService(TeamServiceObserver.class).observeMemberRemove(memberRemoveObs
erver, register);
    }

    /**
     *
     *
     * @param teamId    ID
     * @param accounts
     * @return InvocationFuture ,
     */
    public static void addManagers(String teamId, List<String> accounts,
RequestCallback<List<TeamMember>> callback) {
        NIMClient.getService(TeamService.class)
                .addManagers(teamId, accounts)
                .setCallback(callback);
    }

    /**
     *  <br>
     *
     * @param teamId   ID
     * @param accounts
     * @return InvocationFuture (Normal)
     */
    public static void removeManagers(String teamId, List<String> accounts,
RequestCallback<List<TeamMember>> callback) {
        NIMClient.getService(TeamService.class)
                .removeManagers(teamId, accounts)
                .setCallback(callback);
    }

    /**
     *
     *
```

```
     * @param teamId  ID
     * @param account
     * @param mute    truefalse
     * @return InvocationFuture
     */
    public static void muteTeamMember(String teamId, String account, boolean mute,
RequestCallback<Void> callback) {
        NIMClient.getService(TeamService.class).muteTeamMember(teamId, account,
mute).setCallback(callback);
    }

    /**
     *
     * <p>
     *
     */
    public static void queryMemberList(String teamId, RequestCallback<List<TeamMember>>
callback) {
        NIMClient.getService(TeamService.class).queryMemberList(teamId)
                .setCallback(callback);
    }

    /**
     * ID()
     */
    public static void queryTeamMember(String teamId, String account,
RequestCallbackWrapper<TeamMember> callback) {
        NIMClient.getService(TeamService.class).queryTeamMember(teamId, account)
                .setCallback(callback);
    }

    /**
     * ID()
     */
    public static TeamMember queryTeamMemberBlock(String teamId, String account) {
        return NIMClient.getService(TeamService.class).queryTeamMemberBlock(teamId, account);
    }

    /**
     *
     */
    public static void searchTeam(String teamId, RequestCallback<Team> callback) {
```

```
        NIMClient.getService(TeamService.class).searchTeam(teamId)
                .setCallback(callback);
    }

    /**
     *
     * <p>
     *  TeamService#muteTeamMember
     */
    public static List<TeamMember> queryMutedTeamMembers(String teamId) {
        List<TeamMember> members =
NIMClient.getService(TeamService.class).queryMutedTeamMembers(teamId);
        return members;
    }

    /**
     *
     */
    public static String buildMemberListString(List<String> members, String teamId, String
fromAccount) {
        StringBuilder sb = new StringBuilder();
        for (String account : members) {
            if (!TextUtils.isEmpty(fromAccount) && fromAccount.equals(account)) {
                continue;
            }
            sb.append(getTeamMemberDisplayNameWithYou(teamId, account));
            sb.append(",");
        }
        sb.deleteCharAt(sb.length() - 1);
        return sb.toString();
    }

    /**
     * “”
     */
    public static String getTeamMemberDisplayNameWithYou(String tid, String account) {
        //getTeamMemberDisplayNameYou
        if (account.equals(UserCache.getAccount())) {
            return "";
        }

        return getTeamMemberDisplayNameWithoutMe(tid, account);
```

```
}

/**
 *
 *
 *
 */
public static String getTeamMemberDisplayNameWithoutMe(String tid, String account) {
    String memberNick = getTeamNick(tid, account);
    if (!TextUtils.isEmpty(memberNick)) {
        return memberNick;
    }

    Friend friend = NimFriendSDK.getFriendByAccount(account);
    if (friend != null && !TextUtils.isEmpty(friend.getAlias())) {
        return friend.getAlias();
    }

    NimUserInfo userInfo = NimUserInfoSDK.getUser(account);
    if (userInfo != null && !TextUtils.isEmpty(userInfo.getName())) {
        return userInfo.getName();
    }

    return account;
}

public static String getTeamNick(String tid, String account) {
    Team team = NimTeamSDK.queryTeamBlock(tid);
    if (team != null && team.getType() == TeamTypeEnum.Advanced) {
        TeamMember member = NimTeamSDK.queryTeamMemberBlock(tid, account);
        if (member != null && !TextUtils.isEmpty(member.getTeamNick())) {
            return member.getTeamNick();
        }
    }
    return null;
}

/*==================  start ==================*/
public static String buildMuteTeamNotification(MuteMemberAttachment na, String teamId) {
    StringBuilder sb = new StringBuilder();

    sb.append(buildMemberListString(na.getTargets(), teamId, null));
```

```
        sb.append("");
        sb.append(na.isMute() ? "" : "");

        return sb.toString();
    }

    public static String buildAcceptInviteNotification(MemberChangeAttachment na, String teamId,
String fromAccount) {
        StringBuilder sb = new StringBuilder();

        sb.append(getTeamMemberDisplayNameWithYou(teamId, fromAccount));
        sb.append("  ").append(buildMemberListString(na.getTargets(), teamId, null)).append(" ");

        return sb.toString();
    }

    public static String buildRemoveTeamManagerNotification(MemberChangeAttachment na,
String teamId) {
        StringBuilder sb = new StringBuilder();

        sb.append(buildMemberListString(na.getTargets(), teamId, null));
        sb.append(" ");

        return sb.toString();
    }

    public static String buildAddTeamManagerNotification(MemberChangeAttachment na, String
teamId) {
        StringBuilder sb = new StringBuilder();

        sb.append(buildMemberListString(na.getTargets(), teamId, null));
        sb.append(" ");

        return sb.toString();
    }

    public static String buildTransferOwnerNotification(MemberChangeAttachment na, String
teamId, String fromAccount) {
        StringBuilder sb = new StringBuilder();
        sb.append(getTeamMemberDisplayNameWithYou(teamId, fromAccount));
        sb.append("  ");
        sb.append(buildMemberListString(na.getTargets(), teamId, null));
```

```
        return sb.toString();
    }

    public static String buildManagerPassTeamApplyNotification(MemberChangeAttachment na,
String teamId) {
        StringBuilder sb = new StringBuilder();
        sb.append(" ");
        sb.append(buildMemberListString(na.getTargets(), teamId, null));
        sb.append(" ");

        return sb.toString();
    }

    public static String buildUpdateTeamNotification(UpdateTeamAttachment a, String tid, String
account) {
        StringBuilder sb = new StringBuilder();
        for (Map.Entry<TeamFieldEnum, Object> field : a.getUpdatedFields().entrySet()) {
            if (field.getKey() == TeamFieldEnum.Name) {
                sb.append(" " + field.getValue());
            } else if (field.getKey() == TeamFieldEnum.Introduce) {
                sb.append(" " + field.getValue());
            } else if (field.getKey() == TeamFieldEnum.Announcement) {
                sb.append(getTeamMemberDisplayNameWithYou(tid, account) + " ");
            } else if (field.getKey() == TeamFieldEnum.VerifyType) {
                VerifyTypeEnum type = (VerifyTypeEnum) field.getValue();
                String authen = "";
                if (type == VerifyTypeEnum.Free) {
                    sb.append(authen + "");
                } else if (type == VerifyTypeEnum.Apply) {
                    sb.append(authen + "");
                } else {
                    sb.append(authen + "");
                }
            } else if (field.getKey() == TeamFieldEnum.Extension) {
                sb.append(" " + field.getValue());
            } else if (field.getKey() == TeamFieldEnum.Ext_Server) {
                sb.append("() " + field.getValue());
            } else if (field.getKey() == TeamFieldEnum.ICON) {
                sb.append("");
            } else if (field.getKey() == TeamFieldEnum.InviteMode) {
                sb.append(" " + field.getValue());
```

```
            } else if (field.getKey() == TeamFieldEnum.TeamUpdateMode) {
                sb.append(" " + field.getValue());
            } else if (field.getKey() == TeamFieldEnum.BeInviteMode) {
                sb.append(" " + field.getValue());
            } else if (field.getKey() == TeamFieldEnum.TeamExtensionUpdateMode) {
                sb.append(" " + field.getValue());
            } else if (field.getKey() == TeamFieldEnum.AllMute) {
                TeamAllMuteModeEnum teamAllMuteModeEnum = (TeamAllMuteModeEnum)
field.getValue();
                if (teamAllMuteModeEnum == TeamAllMuteModeEnum.Cancel) {
                    sb.append("");
                } else {
                    sb.append("");
                }
            } else {
                sb.append("" + field.getKey() + " " + field.getValue());
            }
            sb.append("\r\n");
        }
        if (sb.length() < 2) {
            return "";
        }
        return sb.delete(sb.length() - 2, sb.length()).toString();
    }

    public static String buildDismissTeamNotification(String teamId, String fromAccount) {
        return getTeamMemberDisplayNameWithYou(teamId, fromAccount) + " ";
    }

    public static String buildLeaveTeamNotification(String teamId, String fromAccount) {
        String tip;
        Team team = NimTeamSDK.queryTeamBlock(teamId);
        if (team.getType() == TeamTypeEnum.Advanced || team.getType() ==
TeamTypeEnum.Normal) {
            tip = " ";
        } else {
            tip = " ";
        }
        return getTeamMemberDisplayNameWithYou(teamId, fromAccount) + tip;
    }

    public static String buildKickMemberNotification(MemberChangeAttachment na, String teamId,
```

```
String fromAccount) {
        StringBuilder sb = new StringBuilder();
        sb.append(buildMemberListString(na.getTargets(), teamId, null));
        Team team = NimTeamSDK.queryTeamBlock(teamId);
        if (team.getType() == TeamTypeEnum.Advanced || team.getType() ==
TeamTypeEnum.Normal) {
            sb.append(" ");
        } else {
            sb.append(" ");
        }

        return sb.toString();
    }

    public static String buildInviteMemberNotification(MemberChangeAttachment na, String teamId,
String fromAccount) {
        StringBuilder sb = new StringBuilder();
        String selfName = getTeamMemberDisplayNameWithYou(teamId, fromAccount);

        sb.append(selfName);
        sb.append("  ");
        sb.append(buildMemberListString(na.getTargets(), teamId, fromAccount));
        Team team = NimTeamSDK.queryTeamBlock(teamId);
        if (team.getType() == TeamTypeEnum.Advanced || team.getType() ==
TeamTypeEnum.Normal) {
            sb.append(" ");
        } else {
            sb.append(" ");
        }
        return sb.toString();
    }
    /*==================  end ==================*/


    /*==================  start ==================*/
    public static void setShouldShowNickName(String teamId, boolean shouldShowNickName,
RequestCallback<Void> callback) {
        TeamMember member = queryTeamMemberBlock(teamId, UserCache.getAccount());
        if (member != null) {
            Map<String, Object> ext = member.getExtension();
            ext.put(AppConst.MyTeamMemberExt.SHOULD_SHOW_NICK_NAME,
shouldShowNickName);
```

```
            NimTeamSDK.updateMyMemberExtension(teamId, ext, callback);
        }
    }

    public static boolean shouldShowNickName(String teamId) {
        TeamMember member = queryTeamMemberBlock(teamId, UserCache.getAccount());
        if (member == null)
            return false;
        Map<String, Object> ext = member.getExtension();
        Object o = ext.get(AppConst.MyTeamMemberExt.SHOULD_SHOW_NICK_NAME);
        if (o == null)
            return false;
        boolean shouldShowNickName = (boolean) o;
        return shouldShowNickName;
    }
    /*==================  end ==================*/
}
```

94:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\NimUserInfoSDK.java

```
package com.lqr.wechat.nimsdk;

import com.netease.nimlib.sdk.NIMClient;
import com.netease.nimlib.sdk.Observer;
import com.netease.nimlib.sdk.RequestCallback;
import com.netease.nimlib.sdk.RequestCallbackWrapper;
import com.netease.nimlib.sdk.friend.FriendService;
import com.netease.nimlib.sdk.nos.NosService;
import com.netease.nimlib.sdk.uinfo.UserService;
import com.netease.nimlib.sdk.uinfo.UserServiceObserve;
import com.netease.nimlib.sdk.uinfo.constant.UserInfoFieldEnum;
import com.netease.nimlib.sdk.uinfo.model.NimUserInfo;

import java.io.File;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

/**
 * @ CSDN_LQR
```

```
 * @ sdk
 */
public class NimUserInfoSDK {

    /*==================  ==================*/

    /**
     *
     */
    public static List<NimUserInfo> getUsers(List<String> accounts) {
        List<NimUserInfo> users =
NIMClient.getService(UserService.class).getUserInfoList(accounts);
        return users;
    }

    /**
     *
     */
    public static NimUserInfo getUser(String account) {
        NimUserInfo user = NIMClient.getService(UserService.class).getUserInfo(account);
        return user;
    }

    /**
     *
     *
     * @param account
     */
    public static List<NimUserInfo> getUsers(String account) {
        List<NimUserInfo> users = NIMClient.getService(UserService.class).getAllUserInfo();
        return users;
    }

    /**
     *
     * <p>
     *
     */
    public static List<NimUserInfo> getContacts() {
        List<String> accounts = NIMClient.getService(FriendService.class).getFriendAccounts(); //
        List<NimUserInfo> users =
NIMClient.getService(UserService.class).getUserInfoList(accounts); //
```

```
        return users;
    }

    /**
     * ()
     * <p>
     *  SDK
     */
    public static void getUserInfosFormServer(List<String> accounts,
RequestCallback<List<NimUserInfo>> callback) {
        NIMClient.getService(UserService.class).fetchUserInfo(accounts)
                .setCallback(callback);
    }

    /**
     *
     */
    public static void getUserInfoFromServer(String account, RequestCallback<List<NimUserInfo>>
callback) {
        List<String> accounts = new ArrayList<>();
        accounts.add(account);
        getUserInfosFormServer(accounts, callback);
    }

    /*==================  ==================*/

    /**
     *
     */
    public static void uploadFile(File file, String mimeType, RequestCallbackWrapper<String>
callback) {
        NIMClient.getService(NosService.class).upload(file, mimeType)
                .setCallback(callback);
    }

    /**
     *
     * <p>
     *  Map<UserInfoFieldEnum, Object> key value
     *  UserInfoFieldEnum URL
     */
    public static void updateUserInfo(Map<UserInfoFieldEnum, Object> fields,
```

```
RequestCallbackWrapper<Void> callback) {
        NIMClient.getService(UserService.class).updateUserInfo(fields)
                .setCallback(callback);
    }

    /**
     * /
     */
    public static void observeUserInfoUpdate(Observer<List<NimUserInfo>>
userInfoUpdateObserver, boolean register) {
NIMClient.getService(UserServiceObserve.class).observeUserInfoUpdate(userInfoUpdateObserv
er, register);
    }

}
```

95:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\utils\AttachmentStore.java

```
package com.lqr.wechat.nimsdk.utils;

import android.graphics.Bitmap;
import android.text.TextUtils;

import com.lqr.wechat.utils.LogUtils;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

/**
 *
 */
public class AttachmentStore {
    public static long copy(String srcPath, String dstPath) {
    if (TextUtils.isEmpty(srcPath) || TextUtils.isEmpty(dstPath)) {
```

```
    return -1;
    }

        File source = new File(srcPath);
        if (!source.exists()) {
            return -1;
        }

        if (srcPath.equals(dstPath)) {
            return source.length();
        }

        FileChannel fcin = null;
        FileChannel fcout = null;
    try {
            fcin = new FileInputStream(source).getChannel();
            fcout = new FileOutputStream(create(dstPath)).getChannel();
            ByteBuffer tmpBuffer = ByteBuffer.allocateDirect(4096);
            while (fcin.read(tmpBuffer) != -1) {
                tmpBuffer.flip();
                fcout.write(tmpBuffer);
                tmpBuffer.clear();
            }
return source.length();

} catch (FileNotFoundException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
} finally {
try {
if (fcin != null) {
                    fcin.close();
}
if (fcout != null) {
                    fcout.close();
}
} catch (IOException e) {
e.printStackTrace();
}
}
    return -1;
```

```
    }

    public static long getFileLength(String srcPath) {
    if (TextUtils.isEmpty(srcPath)) {
return -1;
}

    File srcFile = new File(srcPath);
    if (!srcFile.exists()) {
return -1;
}

    return srcFile.length();
}

    public static long save(String path, String content) {
        return save(content.getBytes(), path);
    }

    /**
     *
     *
     * @param data
     * @param filePath
     * @return ,-1
     */
    public static long save(byte[] data, String filePath) {
    if (TextUtils.isEmpty(filePath)) {
    return -1;
    }

        File f = new File(filePath);
        if(f.getParentFile() == null) {
        return -1;
        }

        if (!f.getParentFile().exists()) {//
            f.getParentFile().mkdirs();
        }
        try {
            f.createNewFile();
            FileOutputStream fout = new FileOutputStream(f);
```

```
            fout.write(data);
            fout.close();
        } catch (IOException e) {
        e.printStackTrace();
            return -1;
        }
        return f.length();
    }

    public static boolean move(String srcFilePath, String dstFilePath) {
    if (TextUtils.isEmpty(srcFilePath) || TextUtils.isEmpty(dstFilePath)) {
return false;
}

    File srcFile = new File(srcFilePath);
    if (!srcFile.exists() || !srcFile.isFile()) {
return false;
}

        File dstFile = new File(dstFilePath);
        if(dstFile.getParentFile() == null) {
        return false;
        }

        if (!dstFile.getParentFile().exists()) {//
        dstFile.getParentFile().mkdirs();
        }

        return srcFile.renameTo(dstFile);
    }

    public static File create(String filePath) {
        if (TextUtils.isEmpty(filePath)) {
            return null;
        }

        File f = new File(filePath);
        if (!f.getParentFile().exists()) {//
            f.getParentFile().mkdirs();
        }
        try {
            f.createNewFile();
```

```
        return f;
    } catch (IOException e) {
    if(f!=null && f.exists()){
    f.delete();
    }
        return null;
    }
}

/**
 * @param is
 * @param filePath
 * @return -1
 */
public static long save(InputStream is, String filePath) {
    File f = new File(filePath);
    if (!f.getParentFile().exists()) {//
        f.getParentFile().mkdirs();
    }
    FileOutputStream fos = null;
    try {
        f.createNewFile();
        fos = new FileOutputStream(f);
        int read = 0;
        byte[] bytes = new byte[8091];
        while ((read = is.read(bytes)) != -1) {
            fos.write(bytes, 0, read);
        }
        return f.length();
    } catch (IOException e) {
    if(f!=null && f.exists()){
    f.delete();
    }
    LogUtils.e("file", "save is to " + filePath + " failed: " + e.getMessage());
        return -1;
    } finally {
        try {
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
```

```
                if (fos != null) {
                    fos.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    /**
     *
     *
     * @param path
     * @return ,null
     */
    public static byte[] load(String path) {
        try {
        File f = new File(path);
        int unread = (int) f.length();
        int read = 0;
            byte[] buf = new byte[unread]; //
            FileInputStream fin = new FileInputStream(f);
            do {
            int count = fin.read(buf, read, unread);
            read += count;
            unread -= count;
} while (unread != 0);
            fin.close();
            return buf;
        } catch (FileNotFoundException e) {
            return null;
        } catch (IOException e) {
            return null;
        }
    }

    public static String loadAsString(String path) {
        if (isFileExist(path)) {
            byte[] content = load(path);
            return new String(content);
        } else {
            return null;
```

```
        }
    }

    /**
     *
     *
     * @param path
     */
    public static boolean delete(String path) {
        if(TextUtils.isEmpty(path)){
            return false;
        }
        File f = new File(path);
        if (f.exists()) {
            f = renameOnDelete(f);
            return f.delete();
        } else {
return false;
}
    }

    public static void deleteOnExit(String path) {
        if(TextUtils.isEmpty(path)){
            return;
        }
        File f = new File(path);
        if (f.exists()) {
            f.deleteOnExit();
        }
    }

    public static boolean deleteDir(String path) {
        return deleteDir(path, true);
    }

    private static boolean deleteDir(String path, boolean rename) {
        boolean success = true;
        File file = new File(path);
        if (file.exists()) {
            if (rename) {
                file = renameOnDelete(file);
            }
```

```
            File[] list = file.listFiles();
            if (list != null) {
                int len = list.length;
                for (int i = 0; i < len; ++i) {
                    if (list[i].isDirectory()) {
                        deleteDir(list[i].getPath(), false);
                    } else {
                        boolean ret = list[i].delete();
                        if (!ret) {
                            success = false;
                        }
                    }
                }
            }
        } else {
            success = false;
        }
        if (success) {
            file.delete();
        }
        return success;
    }

    // rename before delete to avoid lingering filesystem lock of android
    private static File renameOnDelete(File file) {
        String tmpPath = file.getParent() + "/" + System.currentTimeMillis() + "_tmp";
        File tmpFile = new File(tmpPath);
        if (file.renameTo(tmpFile)) {
            return tmpFile;
        } else {
            return file;
        }
    }

    public static boolean isFileExist(String path) {
if (!TextUtils.isEmpty(path) && new File(path).exists()) {
return true;
}
else {
return false;
}
```

```
}

    public static boolean saveBitmap(Bitmap bitmap, String path, boolean recyle) {
        if (bitmap == null || TextUtils.isEmpty(path)) {
            return false;
        }

        BufferedOutputStream bos = null;
        try {
            FileOutputStream fos = new FileOutputStream(path);
            bos = new BufferedOutputStream(fos);
            bitmap.compress(Bitmap.CompressFormat.JPEG, 80, bos);
            return true;

        } catch (FileNotFoundException e) {
            return false;
        } finally {
            if (bos != null) {
                try {
                    bos.close();
                } catch (IOException e) {
                }
            }
            if (recyle) {
                bitmap.recycle();
            }
        }
    }
}
```

96:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\utils\BitmapDecoder.java

```
package com.lqr.wechat.nimsdk.utils;

import android.annotation.TargetApi;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.media.ThumbnailUtils;
import android.os.Build;
import android.provider.MediaStore;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;

public class BitmapDecoder {
    public static Bitmap decode(InputStream is) {
        BitmapFactory.Options options = new BitmapFactory.Options();

        // RGB_565
        options.inPreferredConfig = Bitmap.Config.RGB_565;

        /**
         * 4.4is
         */
        try {
            if (is.markSupported()) {
                is.reset();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

        try {
            return BitmapFactory.decodeStream(is, null, options);
        } catch (OutOfMemoryError e) {
            e.printStackTrace();
        }

        return null;
    }

    public static Bitmap decodeSampledForDisplay(String pathName) {
        return decodeSampledForDisplay(pathName, true);
    }

    public static Bitmap decodeSampledForDisplay(String pathName, boolean withTextureLimit) {
        float ratio = ImageUtil.MAX_IMAGE_RATIO;
        int[][] reqBounds = new int[][]{
                new int[]{ScreenUtil.screenWidth * 2, ScreenUtil.screenHeight},
                new int[]{ScreenUtil.screenWidth, ScreenUtil.screenHeight * 2},
```

```
            new int[]{(int) (ScreenUtil.screenWidth * 1.414), (int) (ScreenUtil.screenHeight * 1.414)},
    };

    // decode bound
    int[] bound = decodeBound(pathName);
    // pick request bound
    int[] reqBound = pickReqBoundWithRatio(bound, reqBounds, ratio);

    int width = bound[0];
    int height = bound[1];
    int reqWidth = reqBound[0];
    int reqHeight = reqBound[1];

    // calculate sample size
    int sampleSize = SampleSizeUtil.calculateSampleSize(width, height, reqWidth, reqHeight);

    if (withTextureLimit) {
        // adjust sample size
        sampleSize = SampleSizeUtil.adjustSampleSizeWithTexture(sampleSize, width, height);
    }

    int RETRY_LIMIT = 5;
    Bitmap bitmap = decodeSampled(pathName, sampleSize);
    while (bitmap == null && RETRY_LIMIT > 0) {
        sampleSize++;
        RETRY_LIMIT--;
        bitmap = decodeSampled(pathName, sampleSize);
    }

    return bitmap;
}

public static int[] decodeBound(String pathName) {
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(pathName, options);

    return new int[]{options.outWidth, options.outHeight};
}

public static int[] decodeBound(Resources res, int resId) {
    BitmapFactory.Options options = new BitmapFactory.Options();
```

```
    options.inJustDecodeBounds = true;
    BitmapFactory.decodeResource(res, resId, options);

    return new int[]{options.outWidth, options.outHeight};
}

private static int[] pickReqBoundWithRatio(int[] bound, int[][] reqBounds, float ratio) {
    float hRatio = bound[1] == 0 ? 0 : (float) bound[0] / (float) bound[1];
    float vRatio = bound[0] == 0 ? 0 : (float) bound[1] / (float) bound[0];

    if (hRatio >= ratio) {
        return reqBounds[0];
    } else if (vRatio >= ratio) {
        return reqBounds[1];
    } else {
        return reqBounds[2];
    }
}

public static Bitmap decodeSampled(String pathName, int sampleSize) {
    BitmapFactory.Options options = new BitmapFactory.Options();

    // RGB_565
    options.inPreferredConfig = Bitmap.Config.RGB_565;
    // sample size
    options.inSampleSize = sampleSize;

    Bitmap bitmap = null;
    try {
        bitmap = BitmapFactory.decodeFile(pathName, options);
    } catch (OutOfMemoryError e) {
        e.printStackTrace();
        return null;
    }

    return checkInBitmap(bitmap, options, pathName);
}

@TargetApi(Build.VERSION_CODES.HONEYCOMB)
private static Bitmap checkInBitmap(Bitmap bitmap,
                                    BitmapFactory.Options options, String path) {
    boolean honeycomb = Build.VERSION.SDK_INT >=
```

```
Build.VERSION_CODES.HONEYCOMB;
        if (honeycomb && bitmap != options.inBitmap && options.inBitmap != null) {
            options.inBitmap.recycle();
            options.inBitmap = null;
        }

        if (bitmap == null) {
            try {
                bitmap = BitmapFactory.decodeFile(path, options);
            } catch (OutOfMemoryError e) {
                e.printStackTrace();
            }
        }
        return bitmap;
    }

    public static int[] decodeBound(File file) {
        InputStream is = null;
        try {
            is = new FileInputStream(file);
            int[] bound = decodeBound(is);
            return bound;
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } finally {
            if (is != null) {
                try {
                    is.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return new int[]{0, 0};
    }

    public static int[] decodeBound(InputStream is) {
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inJustDecodeBounds = true;
        BitmapFactory.decodeStream(is, null, options);
```

```
        return new int[]{options.outWidth, options.outHeight};
    }

    public static Bitmap decodeSampled(InputStream is, int reqWidth, int reqHeight) {
        BitmapFactory.Options options = new BitmapFactory.Options();

        // RGB_565
        options.inPreferredConfig = Bitmap.Config.RGB_565;
        // sample size
        options.inSampleSize = getSampleSize(is, reqWidth, reqHeight);

        try {
            return BitmapFactory.decodeStream(is, null, options);
        } catch (OutOfMemoryError e) {
            e.printStackTrace();
        }

        return null;
    }

    public static Bitmap decodeSampled(String pathName, int reqWidth, int reqHeight) {
        return decodeSampled(pathName, getSampleSize(pathName, reqWidth, reqHeight));
    }

    public static int getSampleSize(InputStream is, int reqWidth, int reqHeight) {
        // decode bound
        int[] bound = decodeBound(is);

        // calculate sample size
        int sampleSize = SampleSizeUtil.calculateSampleSize(bound[0], bound[1], reqWidth,
reqHeight);

        return sampleSize;
    }

    public static int getSampleSize(String pathName, int reqWidth, int reqHeight) {
        // decode bound
        int[] bound = decodeBound(pathName);

        // calculate sample size
        int sampleSize = SampleSizeUtil.calculateSampleSize(bound[0], bound[1], reqWidth,
reqHeight);
```

```
        return sampleSize;
    }

    /**
     * ****************************** decode resource ***************************************
     */

    public static Bitmap decodeSampled(Resources resources, int resId, int reqWidth, int reqHeight)
{
        return decodeSampled(resources, resId, getSampleSize(resources, resId, reqWidth,
reqHeight));
    }

    public static int getSampleSize(Resources resources, int resId, int reqWidth, int reqHeight) {
        // decode bound
        int[] bound = decodeBound(resources, resId);

        // calculate sample size
        int sampleSize = SampleSizeUtil.calculateSampleSize(bound[0], bound[1], reqWidth,
reqHeight);

        return sampleSize;
    }


    public static Bitmap decodeSampled(Resources res, int resId, int sampleSize) {
        BitmapFactory.Options options = new BitmapFactory.Options();

        // RGB_565
        options.inPreferredConfig = Bitmap.Config.RGB_565;
        // sample size
        options.inSampleSize = sampleSize;

        try {
            return BitmapFactory.decodeResource(res, resId, options);
        } catch (OutOfMemoryError e) {
            e.printStackTrace();
        }

        return null;
    }
```

```
    public static boolean extractThumbnail(String videoPath, String thumbPath) {
        if (!AttachmentStore.isFileExist(thumbPath)) {
            Bitmap thumbnail = ThumbnailUtils.createVideoThumbnail(videoPath,
MediaStore.Images.Thumbnails.MINI_KIND);
            if (thumbnail != null) {
                AttachmentStore.saveBitmap(thumbnail, thumbPath, true);
                return true;
            }
        }
        return false;
    }
}
```

97:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\utils\ExternalStorage.java

```
package com.lqr.wechat.nimsdk.utils;

import android.content.Context;
import android.os.Environment;
import android.os.StatFs;
import android.text.TextUtils;

import java.io.File;
import java.io.IOException;

/** package */
class ExternalStorage {
/**
 *
 */
    private String sdkStorageRoot = null;

private static ExternalStorage instance;

private ExternalStorage() {

}

synchronized public static ExternalStorage getInstance() {
if (instance == null) {
instance = new ExternalStorage();
```

```
}
return instance;
}

    public void init(Context context, String sdkStorageRoot) {
        if (!TextUtils.isEmpty(sdkStorageRoot)) {
            File dir = new File(sdkStorageRoot);
            if (!dir.exists()) {
                dir.mkdirs();
            }
            if (dir.exists() && !dir.isFile()) {
                this.sdkStorageRoot = sdkStorageRoot;
                if (!sdkStorageRoot.endsWith("/")) {
                    this.sdkStorageRoot = sdkStorageRoot + "/";
                }
            }
        }

        if (TextUtils.isEmpty(this.sdkStorageRoot)) {
            loadStorageState(context);
        }

        createSubFolders();
    }

    private void loadStorageState(Context context) {
        String externalPath = Environment.getExternalStorageDirectory().getPath();
        this.sdkStorageRoot = externalPath + "/" + context.getPackageName() + "/";
    }

private void createSubFolders() {
boolean result = true;
File root = new File(sdkStorageRoot);
if (root.exists() && !root.isDirectory()) {
root.delete();
}
for (StorageType storageType : StorageType.values()) {
result &= makeDirectory(sdkStorageRoot + storageType.getStoragePath());
}
if (result) {
createNoMediaFile(sdkStorageRoot);
}
```

```
}

/**
 *
 *
 * @param path
 * @return
 */
private boolean makeDirectory(String path) {
File file = new File(path);
boolean exist = file.exists();
if (!exist) {
exist = file.mkdirs();
}
return exist;
}

protected static String NO_MEDIA_FILE_NAME = ".nomedia";

private void createNoMediaFile(String path) {
File noMediaFile = new File(path + "/" + NO_MEDIA_FILE_NAME);
try {
if (!noMediaFile.exists()) {
noMediaFile.createNewFile();
}
} catch (IOException e) {
e.printStackTrace();
}
}

/**
 *
 *
 * @param fileName
 *            .
 * @return
 */
public String getWritePath(String fileName, StorageType fileType) {
return pathForName(fileName, fileType, false, false);
}

private String pathForName(String fileName, StorageType type, boolean dir,
```

```
   boolean check) {
String directory = getDirectoryByDirType(type);
StringBuilder path = new StringBuilder(directory);

if (!dir) {
path.append(fileName);
}

String pathString = path.toString();
File file = new File(pathString);

if (check) {
if (file.exists()) {
if ((dir && file.isDirectory())
|| (!dir && !file.isDirectory())) {
return pathString;
}
}

return "";
} else {
return pathString;
}
}

/**
 *
 *
 * @param fileType
 * @return
 */
public String getDirectoryByDirType(StorageType fileType) {
return sdkStorageRoot + fileType.getStoragePath();
}

/**
 *
 * @param fileName
    * @param fileType
 * @return
 */
public String getReadPath(String fileName, StorageType fileType) {
```

```
        if (TextUtils.isEmpty(fileName)) {
            return "";
        }

        return pathForName(fileName, fileType, false, true);
    }

    public boolean isSdkStorageReady() {
        String externalRoot = Environment.getExternalStorageDirectory().getAbsolutePath();
        if (this.sdkStorageRoot.startsWith(externalRoot)) {
            return Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED);
        } else {
            return true;
        }
    }

/**
 *
 * @return
 */
    public long getAvailableExternalSize() {
return getResidualSpace(sdkStorageRoot);
}

    /**
     *
     * @param directoryPath
     * @return
     */
    private long getResidualSpace(String directoryPath) {
        try {
            StatFs sf = new StatFs(directoryPath);
            long blockSize = sf.getBlockSize();
            long availCount = sf.getAvailableBlocks();
            long availCountByte = availCount * blockSize;
            return availCountByte;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return 0;
    }
}
```

98:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\utils\ImageUtil.java

```
package com.lqr.wechat.nimsdk.utils;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.Bitmap.CompressFormat;
import android.graphics.Bitmap.Config;
import android.graphics.Matrix;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.media.ExifInterface;
import android.text.TextUtils;

import com.lqr.wechat.R;
import com.lqr.wechat.utils.FileUtils;
import com.lqr.wechat.utils.LogUtils;
import com.lqr.wechat.utils.UIUtils;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;

public class ImageUtil {
    public static class ImageSize {
        public int width = 0;
        public int height = 0;

        public ImageSize(int width, int height) {
            this.width = width;
            this.height = height;
        }
    }

    public final static float MAX_IMAGE_RATIO = 5f;

    public static Bitmap getDefaultBitmapWhenGetFail() {
        try {
```

```
            return getBitmapImmutableCopy(UIUtils.getResource(),
R.mipmap.nim_image_download_failed);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    public static final Bitmap getBitmapImmutableCopy(Resources res, int id) {
        return getBitmap(res.getDrawable(id)).copy(Config.RGB_565, false);
    }

    public static final Bitmap getBitmap(Drawable dr) {
        if (dr == null) {
            return null;
        }

        if (dr instanceof BitmapDrawable) {
            return ((BitmapDrawable) dr).getBitmap();
        }

        return null;
    }

    public static Bitmap rotateBitmapInNeeded(String path, Bitmap srcBitmap) {
        if (TextUtils.isEmpty(path) || srcBitmap == null) {
            return null;
        }

        ExifInterface localExifInterface;
        try {
            localExifInterface = new ExifInterface(path);
            int rotateInt = localExifInterface.getAttributeInt(
                    ExifInterface.TAG_ORIENTATION,
                    ExifInterface.ORIENTATION_NORMAL);
            float rotate = getImageRotate(rotateInt);
            if (rotate != 0) {
                Matrix matrix = new Matrix();
                matrix.postRotate(rotate);
                Bitmap dstBitmap = Bitmap.createBitmap(srcBitmap, 0, 0,
                        srcBitmap.getWidth(), srcBitmap.getHeight(), matrix,
                        false);
```

```
            if (dstBitmap == null) {
                return srcBitmap;
            } else {
                if (srcBitmap != null && !srcBitmap.isRecycled()) {
                    srcBitmap.recycle();
                }
                return dstBitmap;
            }
        } else {
            return srcBitmap;
        }
    } catch (IOException e) {
        e.printStackTrace();
        return srcBitmap;
    }
}

/**
 *
 *
 * @param rotate
 * @return
 */
public static float getImageRotate(int rotate) {
    float f;
    if (rotate == 6) {
        f = 90.0F;
    } else if (rotate == 3) {
        f = 180.0F;
    } else if (rotate == 8) {
        f = 270.0F;
    } else {
        f = 0.0F;
    }

    return f;
}

public static int getImageMaxEdge() {
    return (int) (165.0 / 320.0 * ScreenUtil.screenWidth);
}
```

```
    public static int getImageMinEdge() {
        return (int) (76.0 / 320.0 * ScreenUtil.screenWidth);
    }

    public static String makeThumbnail(Context context, File imageFile) {
        String thumbFilePath = StorageUtils.getWritePath(imageFile.getName(),
                StorageType.TYPE_THUMB_IMAGE);
        File thumbFile = AttachmentStore.create(thumbFilePath);
        if (thumbFile == null) {
            return null;
        }

        boolean result = scaleThumbnail(
                imageFile,
                thumbFile,
                getImageMaxEdge(),
                getImageMinEdge(),
                CompressFormat.JPEG,
                60);
        if (!result) {
            AttachmentStore.delete(thumbFilePath);
            return null;
        }

        return thumbFilePath;
    }

    public static Boolean scaleThumbnail(File srcFile, File dstFile, int dstMaxWH, int dstMinWH,
CompressFormat compressFormat, int quality) {
        Boolean bRet = false;
        Bitmap srcBitmap = null;
        Bitmap dstBitmap = null;
        BufferedOutputStream bos = null;

        try {
            int[] bound = BitmapDecoder.decodeBound(srcFile);
            ImageSize size = getThumbnailDisplaySize(bound[0], bound[1], dstMaxWH, dstMinWH);
            srcBitmap = BitmapDecoder.decodeSampled(srcFile.getPath(), size.width, size.height);

            //
            ExifInterface localExifInterface = new ExifInterface(srcFile.getAbsolutePath());
            int rotateInt = localExifInterface.getAttributeInt(
```

```
            ExifInterface.TAG_ORIENTATION,
            ExifInterface.ORIENTATION_NORMAL);
    float rotate = getImageRotate(rotateInt);

    Matrix matrix = new Matrix();
    matrix.postRotate(rotate);

    float inSampleSize = 1;

    if (srcBitmap.getWidth() >= dstMinWH && srcBitmap.getHeight() <= dstMaxWH
            && srcBitmap.getWidth() >= dstMinWH && srcBitmap.getHeight() <= dstMaxWH) {
        //srcBitmap
    } else {
        if (srcBitmap.getWidth() != size.width || srcBitmap.getHeight() != size.height) {
            float widthScale = (float) size.width / (float) srcBitmap.getWidth();
            float heightScale = (float) size.height / (float) srcBitmap.getHeight();

            if (widthScale >= heightScale) {
                size.width = srcBitmap.getWidth();
                size.height /= widthScale;//srcBitmap.getHeight()
                inSampleSize = widthScale;
            } else {
                size.width /= heightScale;//srcBitmap.getWidth()
                size.height = srcBitmap.getHeight();
                inSampleSize = heightScale;
            }
        }
    }

    matrix.postScale(inSampleSize, inSampleSize);

    if (rotate == 0 && inSampleSize == 1) {
        dstBitmap = srcBitmap;
    } else {
        dstBitmap = Bitmap.createBitmap(srcBitmap, 0, 0, size.width, size.height, matrix, true);
    }

    bos = new BufferedOutputStream(new FileOutputStream(dstFile));
    dstBitmap.compress(compressFormat, quality, bos);
    bos.flush();
    bRet = true;
} catch (OutOfMemoryError e) {
```

```
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (bos != null) {
                try {
                    bos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            if (srcBitmap != null && !srcBitmap.isRecycled()) {
                srcBitmap.recycle();
                srcBitmap = null;
            }

            if (dstBitmap != null && !dstBitmap.isRecycled()) {
                dstBitmap.recycle();
                dstBitmap = null;
            }
        }
        return bRet;
    }

    public static ImageSize getThumbnailDisplaySize(float srcWidth, float srcHeight, float
dstMaxWH, float dstMinWH) {
        if (srcWidth <= 0 || srcHeight <= 0) { // bounds check
            return new ImageSize((int) dstMinWH, (int) dstMinWH);
        }

        float shorter;
        float longer;
        boolean widthIsShorter;

        //store
        if (srcHeight < srcWidth) {
            shorter = srcHeight;
            longer = srcWidth;
            widthIsShorter = false;
        } else {
            shorter = srcWidth;
```

```
            longer = srcHeight;
            widthIsShorter = true;
        }

        if (shorter < dstMinWH) {
            float scale = dstMinWH / shorter;
            shorter = dstMinWH;
            if (longer * scale > dstMaxWH) {
                longer = dstMaxWH;
            } else {
                longer *= scale;
            }
        } else if (longer > dstMaxWH) {
            float scale = dstMaxWH / longer;
            longer = dstMaxWH;
            if (shorter * scale < dstMinWH) {
                shorter = dstMinWH;
            } else {
                shorter *= scale;
            }
        }

        //restore
        if (widthIsShorter) {
            srcWidth = shorter;
            srcHeight = longer;
        } else {
            srcWidth = longer;
            srcHeight = shorter;
        }

        return new ImageSize((int) srcWidth, (int) srcHeight);
    }

    public static File getScaledImageFileWithMD5(File imageFile, String mimeType) {
        String filePath = imageFile.getPath();

        if (!isInvalidPictureFile(mimeType)) {
            LogUtils.i("ImageUtil", "is invalid picture file");
            return null;
        }
```

```
        String tempFilePath = getTempFilePath(FileUtils.getExtensionName(filePath));
        File tempImageFile = AttachmentStore.create(tempFilePath);
        if (tempImageFile == null) {
            return null;
        }

        CompressFormat compressFormat = CompressFormat.JPEG;
        //
        int maxWidth = 720;
        int quality = 60;

        if (ImageUtil.scaleImage(imageFile, tempImageFile, maxWidth, compressFormat, quality)) {
            return tempImageFile;
        } else {
            return null;
        }
    }

    private static String getTempFilePath(String extension) {
        return StorageUtils.getWritePath(
                UIUtils.getContext(),
                "temp_image_" + StringUtil.get36UUID() + "." + extension,
                StorageType.TYPE_TEMP);
    }

    public static Boolean scaleImage(File srcFile, File dstFile, int dstMaxWH, CompressFormat
compressFormat, int quality) {
        Boolean success = false;

        try {
            int inSampleSize = SampleSizeUtil.calculateSampleSize(srcFile.getAbsolutePath(),
dstMaxWH * dstMaxWH);
            Bitmap srcBitmap = BitmapDecoder.decodeSampled(srcFile.getPath(), inSampleSize);
            if (srcBitmap == null) {
                return success;
            }

            //
            ExifInterface localExifInterface = new ExifInterface(srcFile.getAbsolutePath());
            int rotateInt = localExifInterface.getAttributeInt(ExifInterface.TAG_ORIENTATION,
ExifInterface.ORIENTATION_NORMAL);
            float rotate = getImageRotate(rotateInt);
```

```
        Bitmap dstBitmap;
        float scale = (float) Math.sqrt(((float) dstMaxWH * (float) dstMaxWH) / ((float)
srcBitmap.getWidth() * (float) srcBitmap.getHeight()));
        if (rotate == 0f && scale >= 1) {
            dstBitmap = srcBitmap;
        } else {
            try {
                Matrix matrix = new Matrix();
                if (rotate != 0) {
                    matrix.postRotate(rotate);
                }
                if (scale < 1) {
                    matrix.postScale(scale, scale);
                }
                dstBitmap = Bitmap.createBitmap(srcBitmap, 0, 0, srcBitmap.getWidth(),
srcBitmap.getHeight(), matrix, true);
            } catch (OutOfMemoryError e) {
                BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(dstFile));
                srcBitmap.compress(compressFormat, quality, bos);
                bos.flush();
                bos.close();
                success = true;

                if (!srcBitmap.isRecycled())
                    srcBitmap.recycle();
                srcBitmap = null;

                return success;
            }
        }

        BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream(dstFile));
        dstBitmap.compress(compressFormat, quality, bos);
        bos.flush();
        bos.close();
        success = true;

        if (!srcBitmap.isRecycled())
            srcBitmap.recycle();
        srcBitmap = null;
```

```
            if (!dstBitmap.isRecycled())
                dstBitmap.recycle();
            dstBitmap = null;
        } catch (Exception e) {
            e.printStackTrace();
        } catch (OutOfMemoryError e) {
            e.printStackTrace();
        }
        return success;
    }

    public static ImageSize getThumbnailDisplaySize(int maxSide, int minSide, String imagePath) {
        int[] bound = BitmapDecoder.decodeBound(imagePath);
        ImageSize imageSize = getThumbnailDisplaySize(bound[0], bound[1], maxSide, minSide);
        return imageSize;
    }

    public static int[] getBoundWithLength(int maxSide, Object imageObject, boolean
resizeToDefault) {
        int width = -1;
        int height = -1;

        int[] bound;
        if (String.class.isInstance(imageObject)) {
            bound = BitmapDecoder.decodeBound((String) imageObject);
            width = bound[0];
            height = bound[1];
        } else if (Integer.class.isInstance(imageObject)) {
            bound = BitmapDecoder.decodeBound(UIUtils.getResource(), (Integer) imageObject);
            width = bound[0];
            height = bound[1];
        } else if (InputStream.class.isInstance(imageObject)) {
            bound = BitmapDecoder.decodeBound((InputStream) imageObject);
            width = bound[0];
            height = bound[1];
        }

        int defaultWidth = maxSide;
        int defaultHeight = maxSide;
        if (width <= 0 || height <= 0) {
            width = defaultWidth;
```

```
            height = defaultHeight;
        } else if (resizeToDefault) {
            if (width > height) {
                height = (int) (defaultWidth * ((float) height / (float) width));
                width = defaultWidth;
            } else {
                width = (int) (defaultHeight * ((float) width / (float) height));
                height = defaultHeight;
            }
        }

        return new int[]{width, height};
    }

    /**
     *
     *
     * @return
     */
    public static Bitmap getBitmapFromDrawableRes(int res) {
        try {
            return getBitmapImmutableCopy(UIUtils.getResource(), res);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    public static boolean isInvalidPictureFile(String mimeType) {
        String lowerCaseFilepath = mimeType.toLowerCase();
        return (lowerCaseFilepath.contains("jpg") || lowerCaseFilepath.contains("jpeg")
                || lowerCaseFilepath.toLowerCase().contains("png") ||
lowerCaseFilepath.toLowerCase().contains("bmp") || lowerCaseFilepath
                .toLowerCase().contains("gif"));
    }
}
```

99:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\utils\SampleSizeUtil.java

```
package com.lqr.wechat.nimsdk.utils;

import android.opengl.GLES10;
```

```
public class SampleSizeUtil {

public static int calculateSampleSize(String imagePath, int totalPixel) {
int[] bound = BitmapDecoder.decodeBound(imagePath);
return calculateSampleSize(bound[0], bound[1], totalPixel);
}

public static int calculateSampleSize(int width, int height, int totalPixel) {
    int ratio = 1;

        if (width > 0 && height > 0) {
        ratio = (int) Math.sqrt((float) (width * height) / totalPixel);
        if (ratio < 1) {
        ratio = 1;
        }
        }

        return ratio;
}

/**
 * Calculate an inSampleSize for use in a {@link android.graphics.BitmapFactory.Options}
 * object when decoding bitmaps using the decode* methods from
 * {@link android.graphics.BitmapFactory}. This implementation calculates the closest
 * inSampleSize that will result in the final decoded bitmap having a width
 * and height equal to or larger than the requested width and height. This
 * implementation does not ensure a power of 2 is returned for inSampleSize
 * which can be faster when decoding but results in a larger bitmap which
 * isn't as useful for caching purposes.
 *
 * @param width
 * @param height
 * @param reqWidth
 * @param reqHeight
 * @return
 */
public static int calculateSampleSize(int width, int height, int reqWidth, int reqHeight) {
// can't proceed
if (width <= 0 || height <= 0) {
return 1;
}
```

```
// can't proceed
if (reqWidth <= 0 && reqHeight <= 0) {
return 1;
} else if (reqWidth <= 0) {
reqWidth = (int) (width * reqHeight / (float)height + 0.5f) ;
} else if (reqHeight <= 0) {
reqHeight = (int) (height * reqWidth / (float)width + 0.5f);
}

int inSampleSize = 1;

if (height > reqHeight || width > reqWidth) {
// Calculate ratios of height and width to requested height and width
final int heightRatio = Math.round((float) height / (float) reqHeight);
final int widthRatio = Math.round((float) width / (float) reqWidth);

// Choose the smallest ratio as inSampleSize value, this will
// guarantee a final image
// with both dimensions larger than or equal to the requested height and width.
inSampleSize = heightRatio < widthRatio ? heightRatio : widthRatio;
if (inSampleSize == 0) {
inSampleSize = 1;
}

// This offers some additional logic in case the image has a strange
// aspect ratio. For example, a panorama may have a much larger
// width than height. In these cases the total pixels might still
// end up being too large to fit comfortably in memory, so we should
// be more aggressive with sample down the image (=larger
// inSampleSize).

final float totalPixels = width * height;

// Anything more than 2x the requested pixels we'll sample down
// further
final float totalReqPixelsCap = reqWidth * reqHeight * 2;

while (totalPixels / (inSampleSize * inSampleSize) > totalReqPixelsCap) {
inSampleSize++;
}
}
```

```
return inSampleSize;
}

public static final int adjustSampleSizeWithTexture(int sampleSize, int width, int height) {
int textureSize = getTextureSize();

if ((textureSize > 0) && ((width > sampleSize) || (height > sampleSize))) {
while ((width / (float)sampleSize) > textureSize || (height / (float)sampleSize) > textureSize) {
sampleSize++;
}

// 2
sampleSize = SampleSizeUtil.roundup2n(sampleSize);
}

return sampleSize;
}

private static int textureSize = 0;
//static
public static final int getTextureSize() {
if (textureSize > 0) {
return textureSize;
}

int[] params = new int[1];
GLES10.glGetIntegerv(GLES10.GL_MAX_TEXTURE_SIZE, params, 0);
textureSize = params[0];

return textureSize;
}

// x2
private static final int roundup2n(int x) {
if ((x & (x - 1)) == 0) {
return x;
}
int pos = 0;
while (x > 0) {
x >>= 1;
++pos;
}
```

```
return 1 << pos;
}
}
```

100:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\utils\ScreenUtil.java

```
package com.lqr.wechat.nimsdk.utils;

import android.content.Context;
import android.content.res.Resources;
import android.util.DisplayMetrics;
import android.util.Log;

import com.lqr.wechat.utils.UIUtils;

import java.lang.reflect.Field;

public class ScreenUtil {
    private static final String TAG = "Demo.ScreenUtil";

    private static double RATIO = 0.85;

    public static int screenWidth;
    public static int screenHeight;
    public static int screenMin;//
    public static int screenMax;//

    public static float density;
    public static float scaleDensity;
    public static float xdpi;
    public static float ydpi;
    public static int densityDpi;

    public static int dialogWidth;
    public static int statusbarheight;
    public static int navbarheight;

    static {
        init(UIUtils.getContext());
    }

    public static int dip2px(float dipValue) {
```

```
        return (int) (dipValue * density + 0.5f);
    }

    public static int px2dip(float pxValue) {
        return (int) (pxValue / density + 0.5f);
    }

    public static int sp2px(float spValue) {
        return (int) (spValue * scaleDensity + 0.5f);
    }

    public static int getDialogWidth() {
        dialogWidth = (int) (screenMin * RATIO);
        return dialogWidth;
    }

    public static void init(Context context) {
        if (null == context) {
            return;
        }
        DisplayMetrics dm = context.getApplicationContext().getResources().getDisplayMetrics();
        screenWidth = dm.widthPixels;
        screenHeight = dm.heightPixels;
        screenMin = (screenWidth > screenHeight) ? screenHeight : screenWidth;
        density = dm.density;
        scaleDensity = dm.scaledDensity;
        xdpi = dm.xdpi;
        ydpi = dm.ydpi;
        densityDpi = dm.densityDpi;

        Log.d(TAG, "screenWidth=" + screenWidth + " screenHeight=" + screenHeight + " density="
+ density);
    }

    public static int getDisplayWidth() {
        if (screenWidth == 0) {
            GetInfo(UIUtils.getContext());
        }
        return screenWidth;
    }

    public static int getDisplayHeight() {
```

```
        if (screenHeight == 0) {
            GetInfo(UIUtils.getContext());
        }
        return screenHeight;
    }

    public static void GetInfo(Context context) {
        if (null == context) {
            return;
        }
        DisplayMetrics dm = context.getApplicationContext().getResources().getDisplayMetrics();
        screenWidth = dm.widthPixels;
        screenHeight = dm.heightPixels;
        screenMin = (screenWidth > screenHeight) ? screenHeight : screenWidth;
        screenMax = (screenWidth < screenHeight) ? screenHeight : screenWidth;
        density = dm.density;
        scaleDensity = dm.scaledDensity;
        xdpi = dm.xdpi;
        ydpi = dm.ydpi;
        densityDpi = dm.densityDpi;
        statusbarheight = getStatusBarHeight(context);
        navbarheight = getNavBarHeight(context);
        Log.d(TAG, "screenWidth=" + screenWidth + " screenHeight=" + screenHeight + " density="
+ density);
    }

    public static int getStatusBarHeight(Context context) {
        if (statusbarheight == 0) {
            try {
                Class<?> c = Class.forName("com.android.internal.R$dimen");
                Object o = c.newInstance();
                Field field = c.getField("status_bar_height");
                int x = (Integer) field.get(o);
                statusbarheight = context.getResources().getDimensionPixelSize(x);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        if (statusbarheight == 0) {
            statusbarheight = ScreenUtil.dip2px(25);
        }
        return statusbarheight;
```

```
    }

    public static int getNavBarHeight(Context context) {
        Resources resources = context.getResources();
        int resourceId = resources.getIdentifier("navigation_bar_height", "dimen", "android");
        if (resourceId > 0) {
            return resources.getDimensionPixelSize(resourceId);
        }
        return 0;
    }
}
```

101:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\utils\StorageType.java

```
package com.lqr.wechat.nimsdk.utils;

public enum StorageType {
    TYPE_LOG(DirectoryName.LOG_DIRECTORY_NAME),
    TYPE_TEMP(DirectoryName.TEMP_DIRECTORY_NAME),
    TYPE_FILE(DirectoryName.FILE_DIRECTORY_NAME),
    TYPE_AUDIO(DirectoryName.AUDIO_DIRECTORY_NAME),
    TYPE_IMAGE(DirectoryName.IMAGE_DIRECTORY_NAME),
    TYPE_VIDEO(DirectoryName.VIDEO_DIRECTORY_NAME),
    TYPE_THUMB_IMAGE(DirectoryName.THUMB_DIRECTORY_NAME),
    TYPE_THUMB_VIDEO(DirectoryName.THUMB_DIRECTORY_NAME),
    ;
    private DirectoryName storageDirectoryName;
    private long storageMinSize;

    public String getStoragePath() {
return storageDirectoryName.getPath();
}

public long getStorageMinSize() {
return storageMinSize;
}

StorageType(DirectoryName dirName) {
this(dirName, StorageUtils.THRESHOLD_MIN_SPCAE);
}

StorageType(DirectoryName dirName, long storageMinSize) {
```

```
        this.storageDirectoryName = dirName;
        this.storageMinSize = storageMinSize;
}

    enum DirectoryName {
    AUDIO_DIRECTORY_NAME("audio/"),
        DATA_DIRECTORY_NAME("data/"),
        FILE_DIRECTORY_NAME("file/"),
        LOG_DIRECTORY_NAME("log/"),
        TEMP_DIRECTORY_NAME("temp/"),
        IMAGE_DIRECTORY_NAME("image/"),
        THUMB_DIRECTORY_NAME("thumb/"),
        VIDEO_DIRECTORY_NAME("video/"),
        ;

        private String path;

    public String getPath() {
return path;
}

        private DirectoryName(String path) {
this.path = path;
}
    }
}
```

102:F:\git\android\weixinlook\LQRWeChat\app\src\main\java\com\lqr\wechat\nimsdk\utils\StorageUtils.java

```
package com.lqr.wechat.nimsdk.utils;

import android.content.Context;
import android.os.Build;
import android.os.Environment;
import android.text.TextUtils;

import java.io.File;

/**
 * @ CSDN_LQR
 * @ (ExternalStorageStorageType)
 */
```

```
public class StorageUtils {
    public final static long K = 1024;
    public final static long M = 1024 * 1024;
    //
    private static final long THRESHOLD_WARNING_SPACE = 100 * M;
    //
    public static final long THRESHOLD_MIN_SPCAE = 20 * M;

    public static void init(Context context, String rootPath) {
        ExternalStorage.getInstance().init(context, rootPath);
    }

    /**
     * toast
     *
     * @param fileName
     * @param fileType
     * @return null
     */
    public static String getWritePath(String fileName, StorageType fileType) {
        return getWritePath(null, fileName, fileType, false);
    }

    /**
     *
     *
     * @param fileName
     * @param tip      toast
     * @return null
     */
    private static String getWritePath(Context context, String fileName, StorageType fileType,
boolean tip) {
        String path = ExternalStorage.getInstance().getWritePath(fileName, fileType);
        if (TextUtils.isEmpty(path)) {
            return null;
        }
        File dir = new File(path).getParentFile();
        if (dir != null && !dir.exists()) {
            dir.mkdirs();
        }
        return path;
    }
```

```
    /**
     *
     */
    public static boolean isExternalStorageExist() {
        return ExternalStorage.getInstance().isSdkStorageReady();
    }


    /**
     *
     *
     * @param context
     * @param fileType
     * @param tip      toast
     * @return false: , true: ok
     */
    public static boolean hasEnoughSpaceForWrite(Context context, StorageType fileType,
boolean tip) {
        if (!ExternalStorage.getInstance().isSdkStorageReady()) {
            return false;
        }

        long residual = ExternalStorage.getInstance().getAvailableExternalSize();
        if (residual < fileType.getStorageMinSize()) {
            return false;
        } else if (residual < THRESHOLD_WARNING_SPACE) {
        }

        return true;
    }

    /**
     *
     *
     * @param fileName
     * @param fileType
     * @return
     */
    public static String getReadPath(String fileName, StorageType fileType) {
        return ExternalStorage.getInstance().getReadPath(fileName, fileType);
    }
```

```
    /**
     * toast
     *
     * @param context
     * @param fileName
     * @param fileType
     * @return null
     */
    public static String getWritePath(Context context, String fileName, StorageType fileType) {
        return getWritePath(context, fileName, fileType, true);
    }

    public static String getDirectoryByDirType(StorageType fileType) {
        return ExternalStorage.getInstance().getDirectoryByDirType(fileType);
    }

    public static String getSystemImagePath() {
        if (Build.VERSION.SDK_INT > 7) {
            String picturePath =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES).getAbsol
utePath();
            return picturePath + "/nim/";
        } else {
            String picturePath =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DCIM).getAbsolutePa
th();
            return picturePath + "/nim/";
        }
    }

    public static boolean isInvalidVideoFile(String filePath) {
        return filePath.toLowerCase().endsWith(".3gp")
                || filePath.toLowerCase().endsWith(".mp4");
    }

}
```