第三次作业：

登录、短信和下线模块功能实现

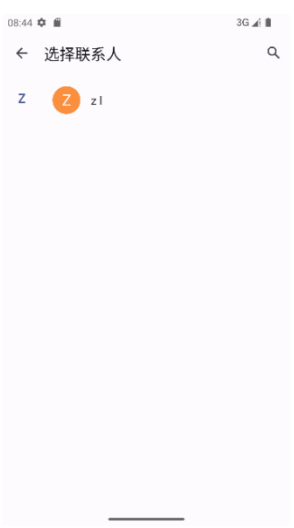
一、登录模块

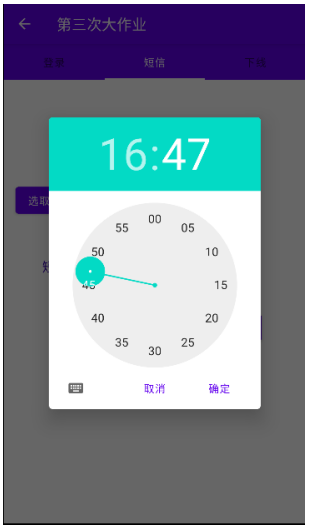
监听登录按钮实现：
1.检验用户名和密码的有效性，置登录状态变量(loginStauts)匹配为 true，不匹配为 false
2.将登录状态通过 SharedPreferences 保存在 xml 文件中
3.清除输入的用户名密码信息(下图中调用方法 clearInfo()实现)
4.跳转到短信模块业务界面

FragmentOne.java

```
//用户名、密码自拟，登录成功后loginStauts设置为true，并写入xml文件
if (username.getText().toString().equals("lz") && password.getText().toString().equals("123")) {
    loginStauts = true;
    Toast.makeText(getContext(), text: "登录成功", Toast.LENGTH_SHORT).show();
    ((MainActivity)getActivity()).viewPager2.setUserInputEnabled(true);
} else {
    loginStauts = false;
    Toast.makeText(getContext(), text: "用户名或密码错误", Toast.LENGTH_SHORT).show();
    ((MainActivity)getActivity()).viewPager2.setUserInputEnabled(false);
}
SharedPreferences sp = getActivity().getSharedPreferences( name: "data", Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sp.edit();
editor.putBoolean("loginStatus", loginStauts);
editor.apply();
clearInfo();
if (loginStauts) ((MainActivity)getActivity()).viewPager2.setCurrentItem(1);
```

二、短信模块





1.读取联系人电话

AndroidManifest.xml 申请权限

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
```

通过 ActivityResultLauncher 动态申请读取联系人权限

```
private void getContact() {
    //选取联系人
    Log.d( tag: "flag",  msg: "选取联系人");
    permissionLauncher1.launch(Manifest.permission.READ_CONTACTS);
}
```

申请具有相关权限后通过 ActivityResultLauncher 执行读取联系人业务操作

```
ActivityResultLauncher<String> permissionLauncher1 = registerForActivityResult(
        new ActivityResultContracts.RequestPermission(),
        new ActivityResultCallback<Boolean>() {
            @Override
            public void onActivityResult(Boolean result) {
                if (result) {
                    contactLauncher.launch( input: null);
                } else {
                    Toast.makeText(getActivity(),  text: "未授予[读取联系人]权限", Toast.LENGTH_SHORT).show();
                }
            }
        }
);
```

通过用户选择的联系人 uri 查询 Id 值，以 id 值为条件来查询对应联系人的电话

```
ActivityResultLauncher<Void> contactLauncher = registerForActivityResult(
        new ActivityResultContracts.PickContact(),
        new ActivityResultCallback<Uri>() {
            @Override
            public void onActivityResult(Uri result) {
                if (result == null) return;
                Cursor cursor = getActivity().getContentResolver()
                        .query(result,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
                if (cursor != null && cursor.moveToFirst()) {
                    int id_index = cursor.getColumnIndex( columnName: "_id");
                    String contactId = cursor.getString(id_index);
                    Cursor cursor2 = getActivity().getContentResolver().query(
                            ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                             projection: null,  selection: "contact_id = ?", new String[]{contactId},  sortOrder: null
                    );
                    if (cursor2 != null && cursor2.moveToFirst()) {
                        String phone0 = "";
                        do {
                            int index = cursor2.getColumnIndex( columnName: "data1");
                            phone0 += cursor2.getString(index);
                            phone0 += ' ';
                        } while (cursor2.moveToNext());
                        phone.setText(phone0.trim());
                        cursor2.close();
                    }
                    cursor.close();
                }
            }
        }
);
```

2.定时发送短信

AndroidManifest.xml 申请权限

```
<uses-permission android:name="android.permission.SEND_SMS" />
```

通过 ActivityResultLauncher 动态申请短信发送权限

```
ActivityResultLauncher<String> permissionLauncher2 = registerForActivityResult(
        new ActivityResultContracts.RequestPermission(),
        new ActivityResultCallback<Boolean>() {
            @Override
            public void onActivityResult(Boolean result) {
                if (!result) {
                    Toast.makeText(getActivity(),  text: "未授予[短信发送]权限", Toast.LENGTH_SHORT).show();
                }
            }
        }
);
```

对联系人电话进行校验，一方面保证不是空值，另一方面，一个联系人可能有多个号码，保证选择唯一号码发送

使用 TimePicker+AlarmManager 实现用户设置倒计时时间触发指令

封装携带电话号码和信息内容数据，可以启动短信发送服务的 intent，通过 PendingIntent.getService 得到延迟后台服务，并在时钟倒计时后启动后台服务发送短信

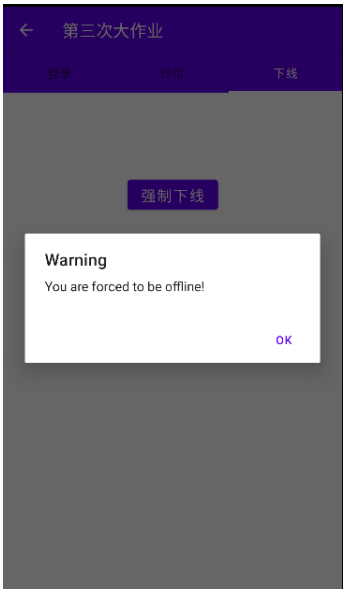
```
private void sendSMS(String s1, String s2) {
    //使用闹钟+SmsManager定时发送短信
    Log.d( tag: "flag",  msg: s1 + " " + s2);
    permissionLauncher2.launch(Manifest.permission.SEND_SMS);
    if (s1.trim().isEmpty()) {
        Toast.makeText(getActivity(),  text: "请选择发送联系人号码", Toast.LENGTH_SHORT).show();
    } else if (s1.trim().length() > 15) {
        Toast.makeText(getActivity(),  text: "请选择唯一有效号码", Toast.LENGTH_SHORT).show();
    } else {
        Calendar calendar = Calendar.getInstance();
        calendar.setTimeZone(TimeZone.getTimeZone("GMT+8"));
        TimePickerDialog tpd = new TimePickerDialog(getActivity(), new TimePickerDialog.OnTimeSetListener() {
            @Override
            public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                Intent intent = new Intent(getActivity(), SmsService.class);
                Bundle bundle = new Bundle();
                bundle.putString("phone", s1);
                bundle.putString("msg", s2);
                intent.putExtras(bundle);
                PendingIntent pendingIntent = PendingIntent.getService(
                        getActivity(),  requestCode: 100, intent, PendingIntent.FLAG_IMMUTABLE);
                Calendar tmp = Calendar.getInstance();
                tmp.setTimeZone(TimeZone.getTimeZone("GMT+8"));
                tmp.set(Calendar.HOUR_OF_DAY, hourOfDay);
                tmp.set(Calendar.MINUTE, minute);
                tmp.set(Calendar.SECOND, 0);
                AlarmManager manager = (AlarmManager) getActivity().getSystemService(Context.ALARM_SERVICE);
                manager.setAndAllowWhileIdle(AlarmManager.RTC_WAKEUP, tmp.getTimeInMillis(), pendingIntent);
            }
        }, calendar.get(Calendar.HOUR_OF_DAY), calendar.get(Calendar.MINUTE),  is24HourView: true);
        tpd.show();
    }
```

短信发送后台服务：
新建 Service SmsService
解析出 intent 携带的电话号码和信息内容
通过 SmsManager 发送短信

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Bundle bundle = intent.getExtras();
    String phone = bundle.getString( key: "phone");
    String msg = bundle.getString( key: "msg");
    SmsManager smsManager = getSystemService(SmsManager.class);
    List<String> list = smsManager.divideMessage(msg);
    for (String m : list) {
        smsManager.sendTextMessage(phone,  scAddress: null, m,  sentIntent: null,  deliveryIntent: null);
    }

    return super.onStartCommand(intent, flags, startId);
}
```

三、下线模块



监听强制下线按钮，发送"com.example.offline"下线广播
FragmentThree.java

```
View.OnClickListener listener = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.btn_offline:
                //强制下线
                Intent intent = new Intent( action: "com.example.offline");
                getActivity().sendBroadcast(intent);
                break;
        }
    }
};
```

下线广播接收器
新建 BroadCastReceiver：OffLineReceiver
在 MainActivity.java 动态注册广播接收器

```
//动态注册广播接收器
br = new OffLineReceiver();
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction("com.example.offline");
registerReceiver(br, intentFilter);
```

接收器代码逻辑
将 xml 文件中保存的登录状态信息 loginStauts 修改为 false
弹出提示框提醒下线消息，设置 builder.setCancelable(false)使对话框不可取消(不会被返回键关闭)
通过 context 重新构造 MainActivity 同时跳转登录页

跳转原因：recreate 方法不同于 finish+startActivity，前者会保存 fragment 的状态信息并在 create 之后恢复。所以 create 之后 viewpager 会被设置为下线按钮页的 fragment，之后触发 viewpager 的 OnPageChangeCallback 回调，因为下线时将登录状态设置为了 false，所以回调中判断到未登录将 fragment 重定向为登录页

OffLineReceiver.java

```
@Override
public void onReceive(Context context, Intent intent) {
    SharedPreferences sharedPreferences = context.getSharedPreferences( name: "data", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putBoolean("loginStatus", false);
    editor.apply();
    AlertDialog.Builder builder = new AlertDialog.Builder(context);
    builder.setTitle("Warning");
    builder.setMessage("You are forced to be offline!");
    builder.setCancelable(false);
    builder.setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            ((MainActivity)context).recreate();
        }
    });
    builder.show();
}
```

四、细节补充

在 MainActivity 的 onCreate 中取消 viewpager 的默认拖拽行为，并在成功登录时恢复。解决未登录时用户左右拖拽出现的业务界面与登录界面来回跳转的问题，提高用户体验

```
viewPager2.setUserInputEnabled(false); //禁止默认页面拖拽，只有成功登录才可拖拽
```

对于 MainActivity 中 TabLayout 的监听程序在未登录时的处理
强制 tablayout 标签选择‘登录’项
设置判断对因回调导致的两次 Toast 只在最后一次回调时执行

```
if (tab.getPosition() == 0) {
    //设置判断的目的是：消除因默认行为回调导致的提醒重复问题
    //在未登录的情况下点击其他模块，会提醒两次请登录，因为第一次回调执行最后一行选择‘登录’标签会再次触发回调
    Toast.makeText( context: MainActivity.this,  text: "请登录", Toast.LENGTH_SHORT).show();
}
viewPager2.setCurrentItem(0);     //未登录时只显示第一个Fragement
tabLayout.selectTab(tabLayout.getTabAt( index: 0));
```

Fragment 与 Activity 之间进行通信

FragmentManager 提供了一个类似于 findViewById()的方法，专门用于从布局文件中获取碎片的实例，代码如下所示：

```
RightFragment rightFragment = (RightFragment) getFragmentManager()
    .findFragmentById(R.id.right_fragment);
```

调用 FragmentManager 的 findFragmentById()方法，可以在活动中得到相应碎片的实例，然后就能轻松地调用碎片里的方法了。

在碎片中可以通过调用 getActivity()方法来得到和当前碎片相关联的活动实例，代码如下所示：

```
MainActivity activity = (MainActivity) getActivity();
```