第三次大作业

主要要求：

在给出的项目模板 ProjectTemplate(采用 Activity+Fragement+ViewPager2 实现)基础上，完成三个 模块功能：登录、短信和下线。

布局的实现（Fragement 和 ViewPager2 的实现）

```
//MainActivity
viewPager2 = findViewById(R.id.viewPager2);
tabLayout = findViewById(R.id.tabLayout);



FragmentManager fragmentManager = getSupportFragmentManager();
fragmentAdapter = new FragmentAdapter(fragmentManager, getLifecycle());
viewPager2.setAdapter(fragmentAdapter);
//FragmentAdapter
public class FragmentAdapter extends FragmentStateAdapter {
    public FragmentAdapter(@NonNull FragmentManager fragmentManager, @NonNull Lifecycle lifecycle) {
        super(fragmentManager, lifecycle);
    }

    @NonNull
    @Override
    public Fragment createFragment(int position) {
        switch (position) {
            case 1:
                return new FragmentTwo();
            case 2:
                return new FragmentThree();
            default:
                return new FragmentOne();
        }
    }
```

【登录模块】FragementOne.java

输入用户名和密码（自拟），登录成功后，其他两个模块才能使用。

要求：将登录状态信息 loginStauts 设置为 true 并保存在 xml 文件中

输入用户名和密码（自拟），登录成功后，其他两个模块才能使用的实现主要代码：

FragmentOne：

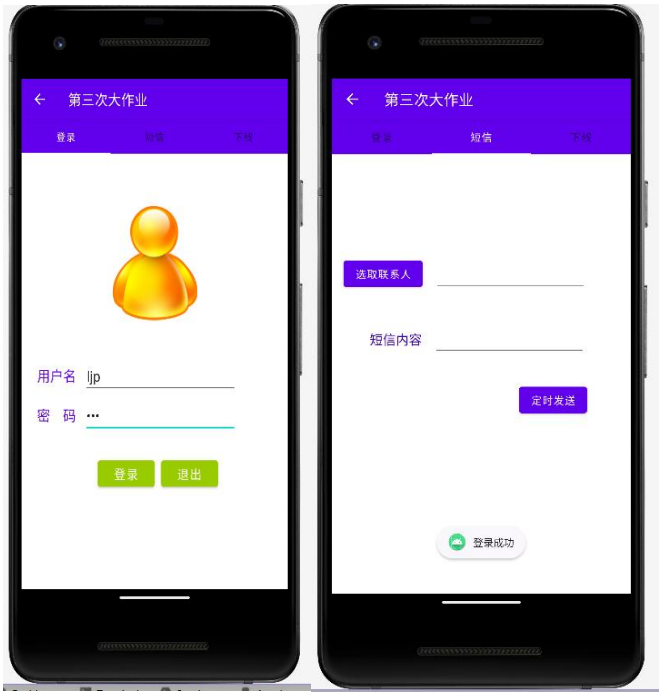
```
boolean loginStauts = false;  //登录状态
View.OnClickListener listener=new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    switch (v.getId()){
      case R.id.btn_login:
        //用户名、密码自拟，登录成功后 loginStauts 设置为 true，并写入 xml 文件
        if (username.getText().toString().equals("ljp") && password.getText().toString().equals("123")) {
          loginStauts = true;
          Toast.makeText(getContext(), "登录成功", Toast.LENGTH_SHORT).show();
           ((MainActivity)getActivity()).viewPager2.setUserInputEnabled(true);
         } else {
          loginStauts = false;
          Toast.makeText(getContext(), "用户名或密码错误", Toast.LENGTH_SHORT).show();
           ((MainActivity)getActivity()).viewPager2.setUserInputEnabled(false);
         }
        SharedPreferences sp = getActivity().getSharedPreferences("data", Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sp.edit();
        editor.putBoolean("loginStatus", loginStauts);
        editor.apply();
        clearInfo();
        if (loginStauts) ((MainActivity)getActivity()).viewPager2.setCurrentItem(1);
        break;
      case R.id.btn_exit:
        getActivity().finish();
        break;
    }
  }
};



public void clearInfo() {
  if (username != null && password != null) {
    username.setText("");
    password.setText("");
  }
}
```

MainActivity：

```
boolean loginStauts = false;       //登录状态，先假设为 true，这样可以看其他两个模块
```

```
tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
  @Override
  public void onTabSelected(TabLayout.Tab tab) {
    SharedPreferences sp = getSharedPreferences("data", MODE_PRIVATE);
    loginStauts = sp.getBoolean("loginStatus", false);
    if(loginStauts){
      viewPager2.setCurrentItem(tab.getPosition());  //已登录情况
     }else{
      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(MainActivity.this, "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(0));
     }

  }

  @Override
  public void onTabUnselected(TabLayout.Tab tab) {

  }

  @Override
  public void onTabReselected(TabLayout.Tab tab) {

  }
});
```

运行情况：



【短信模块】FragementTwo.java

1、读取联系人按钮：点击后获取联系人电话，并放在右侧文本框中显示。

2、定时发送按钮：点击后可设置定时闹钟，实现定时发送短信。

要求：短信发送必须采用 SmsManager + 后台 Service 完成

读取联系人按钮：点击后获取联系人电话，并放在右侧文本框中显示的主要代码实现：

定时发送按钮：点击后可设置定时闹钟，实现定时发送短信主要代码实现：

FragmentTwo：

```
View.OnClickListener listener = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.btn_pick:
                    getContact();
                    break;
                case R.id.btn_sendsms:
                    String s1 = phone.getText().toString();     //联系人电话号码
                    String s2 = msg.getText().toString();     //短信内容
                    sendSMS(s1, s2);
                    break;
            }
        }
```

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

    //注册权限请求 通过 ActivityResultLauncher 动态申请读取联系人权限
    ActivityResultLauncher<String> permissionLauncher1 = registerForActivityResult(
            //RequestPermission 单个权限请求
            new ActivityResultContracts.RequestPermission(),
            //registerForActivityResult() 是 startActivityForResult() + onActivityResult() 的替代，是一种全新的
Activity
            //Results API，它简化了数据回调的写法
            new ActivityResultCallback<Boolean>() {
                @Override
                public void onActivityResult(Boolean result) {
                    if (result) {
                        contactLauncher.launch(null);
                    } else {
                        Toast.makeText(getActivity(), "未授予[读取联系人]权限", Toast.LENGTH_SHORT).show();
                    }
                }
            }
    );

    //查看并选择联系人  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(
            //PickContact 通过 Intent.ACTION_PICK 选择联系人
            new ActivityResultContracts.PickContact(),
            new ActivityResultCallback<Uri>() {
                //registerForActivityResult() 是 startActivityForResult() + onActivityResult() 的替代，是一种全新
的 Activity
                //Results API，它简化了数据回调的写法
                @Override
                public void onActivityResult(Uri result) {
                    if (result == null) return;
                    Cursor cursor = getActivity().getContentResolver()
                            .query(result, null, null, null, null);
                    /*
        第一个参数：电话号码
        第二个参数：运营商，传入 null 就行，系统会自动调用
        第三个参数，短信的内容
```

```
        第四个，第五个参数：短信发送状态的广播，这里不用广播，传入 null
         */
                    if (cursor != null && cursor.moveToFirst()) {
                        int id_index = cursor.getColumnIndex("_id");
                        String contactId = cursor.getString(id_index);
                        Cursor cursor2 = getActivity().getContentResolver().query(
                                ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                                null, "contact_id = ?", new String[]{contactId}, null
                         );
                        if (cursor2 != null && cursor2.moveToFirst()) {
                            String phone0 = "";
                            do {
                                int index = cursor2.getColumnIndex("data1");
                                phone0 += cursor2.getString(index);
                                phone0 += ' ';
                             } while (cursor2.moveToNext());
                            phone.setText(phone0.trim());
                            cursor2.close();
                         }
                        cursor.close();
                    }
                }
            }
    );



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


    //通过 ActivityResultLauncher 动态申请短信发送权限
    private void sendSMS(String s1, String s2) {
```

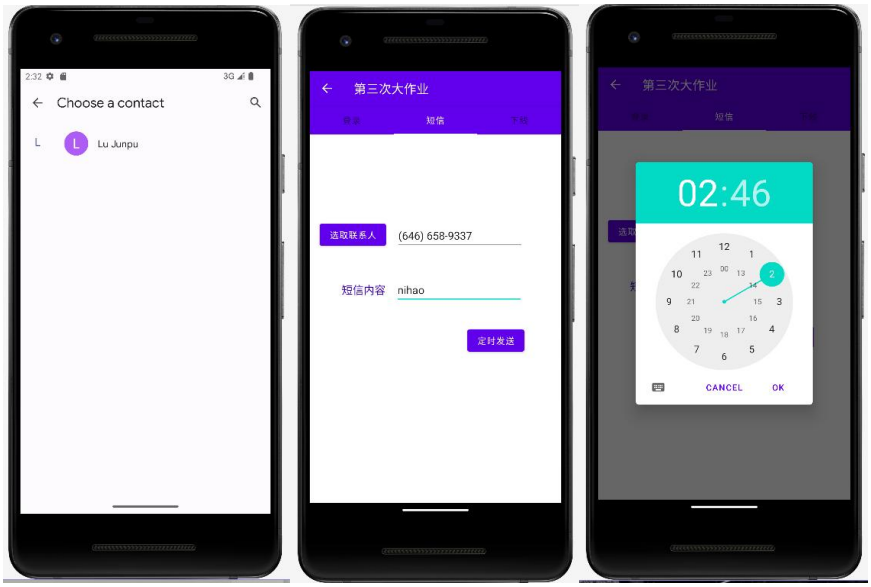
```
    /*     对联系人电话进行校验，一方面保证不是空值，另一方面，一个联系人可能有多
        个号码，保证选择唯一号码发送
        使用 TimePicker+AlarmManager 实现用户设置倒计时时间触发指令
        封装携带电话号码和信息内容数据，可以启动短信发送服务的 intent，通过
        PendingIntent.getService 得到延迟后台服务，并在时钟倒计时后启动后台服
                务发送短信*/

        //使用闹钟+SmsManager 定时发送短信
        Log.d("flag", s1 + " " + s2);
        permissionLauncher2.launch(Manifest.permission.SEND_SMS);
        if (s1.trim().isEmpty()) {
            Toast.makeText(getActivity(), "请选择发送联系人号码", Toast.LENGTH_SHORT).show();
        } else if (s1.trim().length() > 15) {
            Toast.makeText(getActivity(), "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(), 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), true);
            tpd.show();
        }

    }
```

SmsService：

```
/*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*/
```

```
public class SmsService extends Service {
    public SmsService() {
    }

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

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


    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

运行情况：



【下线模块】 FragementThree.java

点击下线按钮：将 xml 文件中保存的登录状态信息 loginStauts 修改为 false，并跳转到登录页面

（FragementOne）。

主 要 代 码 实 现 ：

FragmentThree：

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

    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_three, container, false);

    btn_offline = view.findViewById(R.id.btn_offline);       //下线按钮
    btn_offline.setOnClickListener(listener);

    return view;

}
```

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

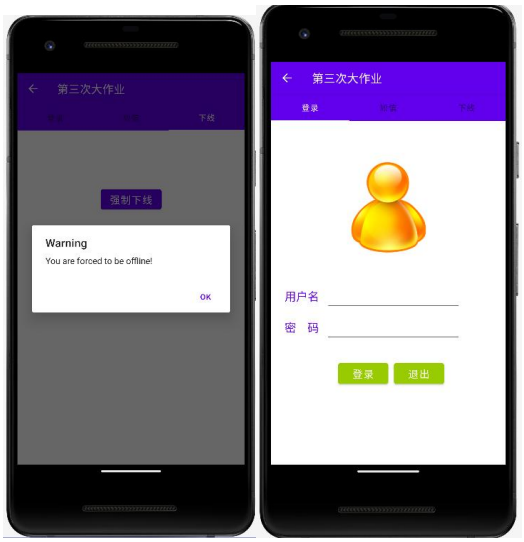
OffLineReceiver：

```
public class OffLineReceiver extends BroadcastReceiver {


    @Override
    public void onReceive(Context context, Intent intent) {
        SharedPreferences sharedPreferences = context.getSharedPreferences("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("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                ((MainActivity)context).recreate();
            }
        });
        builder.show();
    }
}
```

MainActivity：

```
    @Override
    protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver(br);
        SharedPreferences sp = getSharedPreferences("data", MODE_PRIVATE);
        SharedPreferences.Editor editor = sp.edit();
        editor.putBoolean("loginStatus", false);
        editor.apply();
    }
}
```

运行情况：


第三次大作业
下线
强制下线
Warning
You are forced to be offline!
OK
第三次大作业
登录
用户名
密 码
登录
退出