期末考查自写代码行统计

（提示：统计能重点体现工作量和质量的代码！）

（《移动应用开发》至少 1000 行，答辩时展现。）

学号：2040706101　姓名：杨艾琳　专业：计算机科学与技术

我的自写代码行共有__1876__行，具体如下所示：

```
1.    请在下面“2”开始粘贴自写代码行。（粘贴时选择“只粘贴文本”）
2.    package com.example.navigationview;
3.    import androidx.appcompat.app.AppCompatActivity;
4.
5.    import android.content.Intent;
6.    import android.os.Bundle;
7.    import android.view.View;
8.    import android.widget.Button;
9.    import android.widget.EditText;
10.   import android.widget.Toast;
11.
12.   import com.example.navigationview.database.DBHelper;
13.
14.   import java.util.ArrayList;
15.   //登录页面逻辑
16.   public class LoginActivity extends AppCompatActivity {
17.       EditText edt_id,edt_pwd;
18.       @Override
19.       protected void onCreate(Bundle savedInstanceState) {
20.           super.onCreate(savedInstanceState);
21.           setContentView(R.layout.activity_login);
22.           edt_id = findViewById(R.id.edt_uid);
23.           edt_pwd =findViewById(R.id.edt_upwd);
24.           Button btn_login = findViewById(R.id.btn_login);
25.           //登录按键
26.           btn_login.setOnClickListener(new View.OnClickListener() {
27.               @Override
28.               public void onClick(View v) {
29.                   try {
30.                       String userId=edt_id.getText().toString();
31.                       String userPwd=edt_pwd.getText().toString();
32.                       DBHelper dbuserHelper=new
    DBHelper(getApplicationContext());
33.                       User user = dbuserHelper.userlogin(userId,userPwd);
34.
35.                       //登录成功跳转对应类型界面
```

```
                    if(user!=null) {
                        Toast.makeText(getApplicationContext(),
    user.getUserId() + "登录成功", Toast.LENGTH_SHORT).show();
                        Intent intent;
                        ArrayList<User> list = new ArrayList<>();
                        list.add(user);

                        intent = new Intent(getApplicationContext(),
    MainActivity.class);
                        intent.putParcelableArrayListExtra("LoginUser",
    list);
                        startActivity(intent);
                    }else{
                        Toast.makeText(getApplicationContext(),"登录失败,
    密码错误或账号不存在！",Toast.LENGTH_SHORT).show();
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                    Toast.makeText(getApplicationContext(),"数据库异常
    ",Toast.LENGTH_SHORT).show();
                }
            }
        });
        //注册按键
        Button btn_register=findViewById(R.id.btn_register);
        btn_register.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent=new Intent(getApplicationContext(),
    RegisterActivity.class);

                startActivity(intent);
            }
        });
    }
}
package com.example.navigationview;


import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
74.  import android.widget.Toast;
75.
76.  import androidx.appcompat.app.AppCompatActivity;
77.
78.  import com.example.navigationview.database.DBHelper;
79.
80.  import java.util.ArrayList;
81.
82.  //注册页面逻辑
83.  public class RegisterActivity extends AppCompatActivity {
84.      @Override
85.      protected void onCreate(Bundle savedInstanceState) {
86.          super.onCreate(savedInstanceState);
87.          setContentView(R.layout.activity_register);
88.          final EditText edt_rid =findViewById(R.id.edt_rid);
89.          final EditText edt_rpwd =findViewById(R.id.edt_rpwd);
90.          //注册按键
91.          Button btn_registerf=findViewById(R.id.btn_registeruser);
92.          btn_registerf.setOnClickListener(new View.OnClickListener() {
93.              @Override
94.              public void onClick(View v) {
95.                  User user = new User();
96.                  user.setUserId(edt_rid.getText().toString());
97.                  user.setUserPwd(edt_rpwd.getText().toString());
98.
99.                  DBHelper dbUserHelper = new
   DBHelper(getApplicationContext());
100.                 if (dbUserHelper.registerUser(user) > 0) {
101.                     Toast.makeText(getApplicationContext(), "注册成功",
   Toast.LENGTH_SHORT).show();
102.                     Intent intent;
103.                     ArrayList<User> list = new ArrayList<>();
104.                     list.add(user);
105.
106.                     intent = new Intent(getApplicationContext(),
   LoginActivity.class);
107.                     startActivity(intent);
108.
109.                 }else{
110.                     Toast.makeText(getApplicationContext(), "您已经注册过
   此账户，请返回登录", Toast.LENGTH_SHORT).show();
111.                 }
112.             }
113.         });
```

```
    }
}
package com.example.navigationview;
import android.os.Parcel;
import android.os.Parcelable;

public class User implements Parcelable {
    private String userId="";
    private String userPwd="";
    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getUserPwd() {
        return userPwd;
    }

    public void setUserPwd(String userPwd) {
        this.userPwd = userPwd;
    }



    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(this.userId);
        dest.writeString(this.userPwd);
    }

    public User() {
    }

    protected User(Parcel in) {
        this.userId = in.readString();
        this.userPwd = in.readString();
```

```
158.     }
159.
160.     public static final Creator<User> CREATOR = new Creator<User>() {
161.         @Override
162.         public User createFromParcel(Parcel source) {
163.             return new User(source);
164.         }
165.
166.         @Override
167.         public User[] newArray(int size) {
168.             return new User[size];
169.         }
170.     };
171. }
172. package com.example.navigationview.database;
173.
174. import android.annotation.SuppressLint;
175. import android.content.ContentValues;
176. import android.content.Context;
177. import android.database.Cursor;
178. import android.database.SQLException;
179. import android.database.sqlite.SQLiteDatabase;
180. import android.database.sqlite.SQLiteOpenHelper;
181.
182. import com.example.navigationview.R;
183. import com.example.navigationview.User;
184.
185. public class DBHelper extends SQLiteOpenHelper {
186.     public static final String DB_NAME = "MyThings.db";
187.     public static final String TABLE_NAME = "userinfo";
188.     public static final String COLUMN_USERID = "uid";
189.     public static final String COLUMN_USERPWD = "upwd";
190. //    public static final String TABLE_NAME2 = "bill";
191.
192.     //创建数据库语句
193.     private static final String CREATE_TABLE = "create table if not exists
    "
194.             + TABLE_NAME + "(" + COLUMN_USERID + " text not null primary
    key,"
195.             + COLUMN_USERPWD + " text not null)";
196.
197.
198.
199.     public DBHelper(Context context) {
```

```
200.         super(context, DB_NAME, null, 1);
201.     }
202.
203.     //创建数据库方法
204.     @Override
205.     public void onCreate(SQLiteDatabase db) {
206.         try {
207.             db.execSQL(CREATE_TABLE);
208.  //           db.execSQL(CREATE_TABLE2);
209.             //     创建表示类型的表
210.             String sql = "create table typetb(id integer primary key
    autoincrement,typename varchar(10),imageId integer,sImageId integer,kind
    integer)";
211.             db.execSQL(sql);
212.  //           insertType(db);
213.             //创建记账表
214.             sql = "create table accounttb(id integer primary key
    autoincrement,typename varchar(10),sImageId integer,beizhu
    varchar(80),money float," +
215.                     "time varchar(60),year integer,month integer,day
    integer,kind integer)";
216.             db.execSQL(sql);
217.         } catch (SQLException e) {
218.             e.printStackTrace();
219.         }
220.
221.     }
222.
223.
224.     //重置数据库方法(先删表，再建表)
225.     @Override
226.     public void onUpgrade(SQLiteDatabase db, int oldVersion, int
    newVersion) {
227.         db.execSQL("drop table  if exists " + TABLE_NAME);
228.         db.execSQL(CREATE_TABLE);
229.  //       db.execSQL("drop table  if exists " + TABLE_NAME2);
230.  //       db.execSQL(CREATE_TABLE2);
231.     }
232.
233.
234.     //登录方法
235.     @SuppressLint("Range")
236.     public User userlogin(String userId, String userPwd) {
237.         User user = null;
```

```
238.         SQLiteDatabase db = getReadableDatabase();
239.         Cursor cursor = db.query(TABLE_NAME,
240.                 new String[]{COLUMN_USERID, COLUMN_USERPWD},
241.                 COLUMN_USERID + "=? and " + COLUMN_USERPWD + "=?",
242.                 new String[]{userId, userPwd},
243.                 null,
244.                 null,
245.                 null);
246.         if (cursor.getCount() > 0) {
247.             cursor.moveToFirst();
248.             user = new User();
249.
    user.setUserId(cursor.getString(cursor.getColumnIndex(COLUMN_USERID)))
    ;
250.
    user.setUserPwd(cursor.getString(cursor.getColumnIndex(COLUMN_USERPWD))
    );
251.
252.         }
253.         return user;
254.     }
255.
256.     //注册方法
257.     public long registerUser(User user) {
258.         SQLiteDatabase db = getWritableDatabase();
259.         ContentValues contentValues = new ContentValues();
260.         contentValues.put(COLUMN_USERID, user.getUserId());
261.         contentValues.put(COLUMN_USERPWD, user.getUserPwd());
262.         return db.insert(TABLE_NAME, null, contentValues);
263.
264.     }
265. }
266. package com.example.navigationview.database;
267.
268. import android.content.Context;
269.
270. import androidx.room.Database;
271. import androidx.room.Room;
272. import androidx.room.RoomDatabase;
273.
274. @Database(entities ={Thing.class},version = 1,exportSchema = false)
275. public abstract class MyDatabase extends RoomDatabase {
276.     private static final String DATABASE_NAME="MyThings";
277.     private static MyDatabase databaseInstance;
```

```
278.     public static synchronized MyDatabase getInstance(Context context){
279.         if (databaseInstance==null){
280.             databaseInstance=Room.databaseBuilder(context.getApplicationContext(),
         MyDatabase.class,DATABASE_NAME).build();
281.         }
282.         return databaseInstance;
283.     }
284.     public abstract ThingsDao thingsDao();
285. }
286. package com.example.navigationview.database;
287.
288. import androidx.annotation.NonNull;
289. import androidx.room.ColumnInfo;
290. import androidx.room.Entity;
291. import androidx.room.Ignore;
292. import androidx.room.PrimaryKey;
293.
294. @Entity(tableName = "thing")
295. public class Thing {
296.     @NonNull
297.     @PrimaryKey
298.     @ColumnInfo(name = "thingid",typeAffinity = ColumnInfo.TEXT)
299.     public String thingid;
300.     @ColumnInfo(name = "thingname",typeAffinity = ColumnInfo.TEXT)
301.     public String thingname;
302.     @ColumnInfo(name = "publishtime",typeAffinity = ColumnInfo.TEXT)
303.     public String publishtime;
304.     @ColumnInfo(name = "img",typeAffinity = ColumnInfo.BLOB)
305.     public byte[] img;
306.
307.     public Thing(String thingid, String thingname, String publishtime,
         byte[] img){
308.         this.thingid=thingid;
309.         this.thingname=thingname;
310.         this.publishtime=publishtime;
311.         this.img=img;
312.     }
313.     @Ignore
314.     public Thing(String thingname, String publishtime, byte[] img){
315.         this.thingname=thingname;
316.     }
317.
318. }
```

```
package com.example.navigationview.database;

import androidx.lifecycle.LiveData;
import androidx.room.Dao;
import androidx.room.Delete;
import androidx.room.Insert;
import androidx.room.Query;
import androidx.room.Update;

import java.util.List;
@Dao
public interface ThingsDao {
    @Insert
    void insertThing(Thing thing);

    @Delete
    void deleteThing(Thing thing);

    @Update
    void updateThing(Thing thing);

    @Query("SELECT * FROM thing")
    LiveData<List<Thing>> getThingList();
    @Query("SELECT * FROM thing WHERE thingid=:thingid")
    Thing getThingByThingId(String thingid);
}
package com.example.navigationview;

import android.annotation.SuppressLint;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.Toast;
```

```
363.
364.  import androidx.appcompat.app.AlertDialog;
365.  import androidx.appcompat.app.AppCompatActivity;
366.
367.  import java.util.ArrayList;
368.  import java.util.HashMap;
369.  import java.util.Map;
370.
371.  public class ManageActivity extends AppCompatActivity {
372.      private SQLiteDatabase sqLiteDatabase = null;
373.      private int selectId = -1;
374.      EditText edt_date, edt_type, edt_money, edt_state;
375.      TextView tv_test;
376.
377.      private static final String DATABASE_NAME = "MyThings.db";
378.      private static final String TABLE_NAME = "record";
379.      private static final String COLUMN_ID = "id";
380.      private static final String COLUMN_DATE = "date";
381.      private static final String COLUMN_TYPE = "type";
382.      private static final String COLUMN_MONEY = "money";
383.      private static final String COLUMN_STATE = "state";
384.
385.      //创建表
386.      private static final String CREATE_TABLE = "create table if not exists
    " + TABLE_NAME
387.              + "(" + COLUMN_ID + " integer primary key autoincrement," +
    COLUMN_DATE + " text," + COLUMN_TYPE
388.              + " text," + COLUMN_MONEY + " float," + COLUMN_STATE + " text)";
389.      @SuppressLint("Range")
390.      //自定义的查询方法
391.      private void selectData() {
392.          //遍历整个表
393.          String sql = "select * from " + TABLE_NAME ;
394.          //把查询数据封装到 Cursor
395.          Cursor cursor = sqLiteDatabase.rawQuery(sql, null);
396.          ArrayList<Map<String, String>> list = new ArrayList<Map<String,
    String>>();
397.          //用 while 循环遍历 Cursor，再把它分别放到 map 中，最后统一存入 list 中，
    便于调用
398.          while (cursor.moveToNext()) {
399.
400.              int id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID));
401.              String date =
    cursor.getString(cursor.getColumnIndex(COLUMN_DATE));
```

```
            String type =
    cursor.getString(cursor.getColumnIndex(COLUMN_TYPE));
            float money =
    cursor.getFloat(cursor.getColumnIndex(COLUMN_MONEY));
            String state =
    cursor.getString(cursor.getColumnIndex(COLUMN_STATE));

            Map<String, String> map = new HashMap<String, String>();
            map.put("id", String.valueOf(id));
            map.put("date", date);
            map.put("type", type);
            map.put("money", String.valueOf(money));
            map.put("state", state);
            list.add(map);
        }

        //创建 SimpleAdapter
        SimpleAdapter simpleAdapter = new
    SimpleAdapter(getApplicationContext(),
                list,
                R.layout.record_item_layout,
                new String[]{"id", "date", "type", "money", "state"},
                new int[]{R.id.list_id, R.id.list_date, R.id.list_type,
    R.id.list_money, R.id.list_state});
        final ListView listView = findViewById(R.id.recycler_view);
        //绑定适配器
        listView.setAdapter(simpleAdapter);
        //设置 ListView 单击事件
        listView.setOnItemClickListener(new
    AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
    position, long id) {
                ListView tempList = (ListView) parent;
                View mView = tempList.getChildAt(position);
                TextView list_id = mView.findViewById(R.id.list_id);
                TextView list_date = mView.findViewById(R.id.list_date);
                TextView list_type = mView.findViewById(R.id.list_type);
                TextView list_money =
    mView.findViewById(R.id.list_money);
                TextView list_state =
    mView.findViewById(R.id.list_state);

                String rid = list_id.getText().toString();
```

```
437.                String date = list_date.getText().toString();
438.                String type = list_type.getText().toString();
439.                String money = list_money.getText().toString();
440.                String state = list_state.getText().toString();
441.
442.                tv_test.setText(rid);
443.                edt_date.setText(date);
444.                edt_type.setText(type);
445.                edt_money.setText(money);
446.                edt_state.setText(state);
447.                selectId = Integer.parseInt(rid);
448.
449.            }
450.        });
451.    }
452.
453.    @Override
454.    protected void onCreate(Bundle savedInstanceState) {
455.        super.onCreate(savedInstanceState);
456.        setContentView(R.layout.activity_manage);
457.        try {
458.            sqLiteDatabase = openOrCreateDatabase(DATABASE_NAME,
   MODE_PRIVATE, null);
459.            sqLiteDatabase.execSQL(CREATE_TABLE);
460.            //执行查询
461.            selectData();
462.        } catch (SQLException e) {
463.            Toast.makeText(this, "数据库异常!",
   Toast.LENGTH_LONG).show();
464.            e.printStackTrace();
465.        }
466.        tv_test = findViewById(R.id.tv_test);
467.        edt_date = findViewById(R.id.edt_date);
468.        edt_type = findViewById(R.id.edt_type);
469.        edt_money = findViewById(R.id.edt_money);
470.        edt_state = findViewById(R.id.edt_state);
471.
472.        //新增按键
473.        Button btn_add = findViewById(R.id.btn_add);
474.        btn_add.setOnClickListener(new View.OnClickListener() {
475.            @Override
476.            public void onClick(View v) {
477.                if (edt_date.getText().toString().equals("") |
   edt_type.getText().toString().equals("") |
```

```
    edt_money.getText().toString().equals("") |
    edt_state.getText().toString().equals("")) {
                    Toast.makeText(ManageActivity.this, "数据不能为空!",
    Toast.LENGTH_LONG).show();
                    return;
                }

                String date = edt_date.getText().toString();
                String type = edt_type.getText().toString();
                String money = edt_money.getText().toString();
                String state = edt_state.getText().toString();
                //定义添加数据的 sql 语句
                String sql = "insert into " + TABLE_NAME + "(" + COLUMN_DATE
    + "," + COLUMN_TYPE + "," + COLUMN_MONEY + "," + COLUMN_STATE + ") " +
                        "values('" + date + "','" + type + "','" + money +
    "','" + state + "')";
                //执行 sql 语句
                sqLiteDatabase.execSQL(sql);
                Toast.makeText(getApplicationContext(), "添加成功!",
    Toast.LENGTH_LONG).show();
                //刷新显示列表
                selectData();

                //消除数据
                tv_test.setText("");
                edt_date.setText("");
                edt_type.setText("");
                edt_money.setText("");
                edt_state.setText("");
            }
        });

        //修改按键
        Button btn_update = findViewById(R.id.btn_update);
        btn_update.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //无选择提示
                if (selectId == -1) {
                    Toast.makeText(getApplicationContext(), "请选择要修改
    的记录!", Toast.LENGTH_LONG).show();
                    return;
                }
                //判断是否有空数据
```

```
515.                if (edt_date.getText().toString().equals("") |
    edt_type.getText().toString().equals("") |
    edt_money.getText().toString().equals("") |
    edt_state.getText().toString().equals("")) {
516.                    Toast.makeText(getApplicationContext(), "数据不能为
    空!", Toast.LENGTH_LONG).show();
517.                    return;
518.                }
519.
520.                String date = edt_date.getText().toString();
521.                String type = edt_type.getText().toString();
522.                String money = edt_money.getText().toString();
523.                String state = edt_state.getText().toString();
524.                //定义修改数据的 sql 语句
525.                String sql = "update " + TABLE_NAME + " set " + COLUMN_DATE
    + "='" + date + "',type='" + type + "',money='" + money + "',state='" +
    state + "' where id=" + selectId;
526.                //执行 sql 语句
527.                sqLiteDatabase.execSQL(sql);
528.                Toast.makeText(getApplicationContext(), "修改成功",
    Toast.LENGTH_LONG).show();
529.                //刷新显示列表
530.                selectData();
531.                selectId = -1;
532.                //消除数据
533.                tv_test.setText("");
534.                edt_date.setText("");
535.                edt_type.setText("");
536.                edt_money.setText("");
537.                edt_state.setText("");
538.            }
539.        });
540.
541.        //删除按键
542.        Button btn_delete = findViewById(R.id.btn_delete);
543.        btn_delete.setOnClickListener(new View.OnClickListener() {
544.            @Override
545.            public void onClick(View v) {
546.                //无选择提示
547.                if (selectId == -1) {
548.                    Toast.makeText(ManageActivity.this, "请选择要删除的记录
    ", Toast.LENGTH_LONG).show();
549.                    return;
550.                }
```

```
551.
552.                 //定义删除对话框
553.                 AlertDialog dialog = new
    AlertDialog.Builder(ManageActivity.this).setTitle("删除操作")
554.                         .setMessage("是否确定删除所选记录")
555.                         .setPositiveButton("确定", new
    DialogInterface.OnClickListener() {
556.                             @Override
557.                             public void onClick(DialogInterface dialog, int
    which) {
558.                                 //定义删除的 sql 语句
559.                                 String sql = "delete from " + TABLE_NAME +
    " where id=" + selectId;
560.                                 //执行 sql 语句
561.                                 sqLiteDatabase.execSQL(sql);
562.                                 //刷新显示列表
563.                                 Toast.makeText(getApplicationContext(), "
    删除成功", Toast.LENGTH_LONG).show();
564.                                 selectData();
565.                                 selectId = -1;
566.                                 //清除数据
567.                                 tv_test.setText("");
568.                                 edt_date.setText("");
569.                                 edt_type.setText("");
570.                                 edt_money.setText("");
571.                                 edt_state.setText("");
572.                             }
573.                         }).setNegativeButton("取消", new
    DialogInterface.OnClickListener() {
574.                             @Override
575.                             public void onClick(DialogInterface dialog, int
    which) {
576.
577.                             }
578.                         }).create();
579.                 dialog.show();
580.             }
581.         });
582.
583.         //统计页面
584.         //收支统计
585.         ImageView btn_calcmoney=findViewById(R.id.total);
586.         btn_calcmoney.setOnClickListener(new View.OnClickListener() {
587.             @Override
```

```
            public void onClick(View v) {
                Intent intent3=new
   Intent(getApplicationContext(),SearchRecordActivity.class);
                startActivity(intent3);
            }
        });
    }
}
package com.example.navigationview;

import android.annotation.SuppressLint;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

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

//收支记录页面业务逻辑
public class SearchRecordActivity extends AppCompatActivity {
    //定义 spinner 中的数据
    private String[] date_data= {"", "202201", "202202", "202203",
   "202204",
   "202205","202206","202207","202208","202209","202210","202211","202212
   "};
    private String[] type_data = {"", "收入", "支出"};
    Spinner spin_date, spin_type;
    ListView listView;
    TextView tv_show;
    float sum=0;
```

```
628.     //数据库
629.     private String selectDate, selectType;
630.     private static final String DATABASE_NAME = "MyThings.db";
631.     private static final String TABLE_NAME = "record";
632.     private static final String COLUMN_ID = "id";
633.     private static final String COLUMN_DATE = "date";
634.     private static final String COLUMN_TYPE = "type";
635.     private static final String COLUMN_MONEY = "money";
636.     private static final String COLUMN_STATE = "state";
637.     private SQLiteDatabase sqLiteDatabase = null;
638.
639.     private void selectSumMoney() {
640.         //自定义查询的 sql 语句
641.         String sql;
642.         //如果查询时间和查询类型都为空，则查询整个表
643.         if (TextUtils.isEmpty(selectDate) &&
    TextUtils.isEmpty(selectType)) {
644.             sql = "select * from " + TABLE_NAME;
645.             //如果有查询时间，没有查询类型，查询指定内容
646.         } else if (!TextUtils.isEmpty(selectDate) &&
    TextUtils.isEmpty(selectType)) {
647.             sql = "select * from " + TABLE_NAME + " where date='" + selectDate
    + "'";
648.             //如果没有查询时间，有查询类型，查询指定内容
649.         } else if (TextUtils.isEmpty(selectDate)
    && !TextUtils.isEmpty(selectType)) {//如果没有查询时间，有查询类型
650.             sql = "select * from " + TABLE_NAME + " where type='" +
    selectType+"'";
651.         } else {//否则，查询条件都不为空，查询指定内容
652.             sql ="select * from " + TABLE_NAME + " where date='" + selectDate
    + "' and type='" + selectType+"'";
653.         }
654.         Cursor cursor = sqLiteDatabase.rawQuery(sql, null);
655.
656. //       while (cursor.moveToNext()) {
657. //
658. //           float money =
    cursor.getFloat(cursor.getColumnIndex(COLUMN_MONEY));
659. //           sum=sum+money;
660. //
661. //           //list.add(map);
662. //       }
663.         String money2=String.valueOf(sum);
664.         tv_show.setText(money2);
```

```
665.         sum=0;
666.     }
667.     //选择数据
668.     private void selectData() {
669.         //自定义查询的 sql 语句
670.         String sql;
671.         //如果查询时间和查询类型都为空，则查询整个表
672.         if (TextUtils.isEmpty(selectDate) &&
    TextUtils.isEmpty(selectType)) {
673.             sql = "select * from " + TABLE_NAME;
674.             //如果有查询时间，没有查询类型，查询指定内容
675.         } else if (!TextUtils.isEmpty(selectDate) &&
    TextUtils.isEmpty(selectType)) {
676.             sql = "select * from " + TABLE_NAME + " where date='" + selectDate
    + "'";
677.             //如果没有查询时间，有查询类型，查询指定内容
678.         } else if (TextUtils.isEmpty(selectDate)
    && !TextUtils.isEmpty(selectType)) {//如果没有查询时间，有查询类型
679.             sql = "select * from " + TABLE_NAME + " where type='" +
    selectType+"'";
680.         } else {//否则，查询条件都不为空，查询指定内容
681.             sql = "select * from " + TABLE_NAME + " where date='" + selectDate
    + "' and type='" + selectType+"'";
682.         }
683.         //将查询到的数据封装到 Cursor
684.         Cursor cursor = sqLiteDatabase.rawQuery(sql, null);
685.         ArrayList<Map<String, String>> list = new ArrayList<Map<String,
    String>>();
686.         if (cursor.getCount() == 0) {
687.             //查无数据则怒不显示列表
688.             // listView.setVisibility(View.GONE);
689.             Toast.makeText(getApplicationContext(), "无数据",
    Toast.LENGTH_SHORT).show();
690.         } else {
691.             //查有数据则显示列表
692.
693.             listView.setVisibility(View.VISIBLE);
694.
695.             while (cursor.moveToNext()) {
696.                 @SuppressLint("Range")
697.                 int id =
    cursor.getInt(cursor.getColumnIndex(COLUMN_ID));
698.                 @SuppressLint("Range")
```

```
699.                String date =
    cursor.getString(cursor.getColumnIndex(COLUMN_DATE));
700.                @SuppressLint("Range")
701.                String type =
    cursor.getString(cursor.getColumnIndex(COLUMN_TYPE));
702.                @SuppressLint("Range")
703.                float money =
    cursor.getFloat(cursor.getColumnIndex(COLUMN_MONEY));
704.                @SuppressLint("Range")
705.                String state =
    cursor.getString(cursor.getColumnIndex(COLUMN_STATE));
706.                Map<String, String> map = new HashMap<String, String>();
707.                map.put("id", String.valueOf(id));
708.                map.put("date", date);
709.                map.put("type", type);
710.                map.put("money", String.valueOf(money));
711.                map.put("state", state);
712.                list.add(map);
713.            }
714.            //创建 SimpleAdapter
715.            SimpleAdapter simpleAdapter = new
    SimpleAdapter(getApplicationContext(),
716.                    list,
717.                    R.layout.record_item_layout,
718.                    new String[]{"id", "date", "type", "money", "state"},
719.                    new int[]{R.id.list_id, R.id.list_date,
    R.id.list_type, R.id.list_money, R.id.list_state});
720.            listView.setAdapter(simpleAdapter);
721.
722.        }
723.    }
724.
725.    //时间和类别 spinner 点击事件
726.    private void initClick() {
727.        //时间事件
728.        spin_date.setOnItemSelectedListener(new
    AdapterView.OnItemSelectedListener() {
729.            @Override
730.            public void onItemSelected(AdapterView<?> parent, View view,
    int position, long id) {
731.                selectDate = date_data[position];
732.            }
733.
734.            @Override
```

```
735.             public void onNothingSelected(AdapterView<?> parent) {
736.             }
737.         });
738.         //类别事件
739.         spin_type.setOnItemSelectedListener(new
    AdapterView.OnItemSelectedListener() {
740.             @Override
741.             public void onItemSelected(AdapterView<?> parent, View view,
    int position, long id) {
742.                 selectType = type_data[position];
743.             }
744.
745.             @Override
746.             public void onNothingSelected(AdapterView<?> parent) {
747.             }
748.         });
749.
750.         findViewById(R.id.btn_search).setOnClickListener(new
    View.OnClickListener() {
751.             @Override
752.             public void onClick(View v) {
753.                 selectData();
754.             }
755.         });
756.     }
757.
758.     @Override
759.     protected void onCreate(Bundle savedInstanceState) {
760.         super.onCreate(savedInstanceState);
761.         setContentView(R.layout.activity_search_record);
762.         tv_show=findViewById(R.id.tv_show);
763.         try {
764.             //打开数据库，如果是第一次会创建该数据库，模式为 MODE_PRIVATE
765.             sqLiteDatabase = openOrCreateDatabase(DATABASE_NAME,
    MODE_PRIVATE, null);
766.             //执行创建表的 sql 语句，虽然每次都调用，但只有首次才创建表
767.
768.             //执行查询
769.             listView = findViewById(R.id.searchlistview);//绑定列表
770.             selectData();
771.         } catch (SQLException e) {
772.             Toast.makeText(this, "数据库异常!",
    Toast.LENGTH_LONG).show();
773.             e.printStackTrace();
```

```
774.         }
775.
776.         ArrayAdapter<String> adapter = new
777.                ArrayAdapter<String>(this,
   android.R.layout.simple_spinner_item, date_data);
778.
   adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdo
   wn_item);
779.         spin_date = findViewById(R.id.spin_date);
780.         spin_date.setAdapter(adapter);
781.
782.         ArrayAdapter<String> adapter1 = new
783.                ArrayAdapter<String>(this,
   android.R.layout.simple_spinner_item, type_data);
784.
   adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdo
   wn_item);
785.         spin_type = findViewById(R.id.spin_type);
786.         spin_type.setAdapter(adapter1);
787.         initClick();
788.  //       Button btn_calc=findViewById(R.id.btn_calc);
789.  //       btn_calc.setOnClickListener(new View.OnClickListener() {
790.  //           @Override
791.  //           public void onClick(View v) {
792.  //               selectSumMoney();
793.  //
794.  //           }
795.  //       });
796.
797.     }
798.  }
799.  package com.example.navigationview.ui.addthing;
800.
801.  import android.app.AlertDialog;
802.  import android.app.DatePickerDialog;
803.  import android.app.Dialog;
804.  import android.content.DialogInterface;
805.  import android.content.Intent;
806.  import android.graphics.Bitmap;
807.  import android.net.Uri;
808.  import android.os.AsyncTask;
809.  import android.os.Bundle;
810.  import android.view.Gravity;
811.  import android.view.LayoutInflater;
```

```
812.  import android.view.View;
813.  import android.view.ViewGroup;
814.  import android.widget.ArrayAdapter;
815.  import android.widget.AutoCompleteTextView;
816.  import android.widget.Button;
817.  import android.widget.DatePicker;
818.  import android.widget.EditText;
819.  import android.widget.ImageView;
820.
821.  import androidx.annotation.NonNull;
822.  import androidx.fragment.app.Fragment;
823.  import androidx.navigation.NavController;
824.  import androidx.navigation.Navigation;
825.
826.  import com.bumptech.glide.Glide;
827.  import com.bumptech.glide.request.animation.GlideAnimation;
828.  import com.bumptech.glide.request.target.SimpleTarget;
829.  import com.example.navigationview.R;
830.  import com.example.navigationview.database.MyDatabase;
831.  import com.example.navigationview.database.Thing;
832.  import com.linchaolong.android.imagepicker.ImagePicker;
833.
834.  import java.io.ByteArrayOutputStream;
835.  import java.io.File;
836.  import java.util.Calendar;
837.
838.  public class addthingFragment extends Fragment {
839.      // 自动完成文本框
840.      AutoCompleteTextView thingid;
841.      String[] thingids = { "123", "122", "121" };
842.      // 退出按钮
843.      static final int EXIT_DIALOG_ID = 0;
844.      // 出版日期
845.      EditText publishtime;
846.      static final int DATE_DIALOG_ID = 1;
847.      private int mYear;
848.      private int mMonth;
849.      private int mDay;
850.      //picture
851.      private ImagePicker imagePicker = new ImagePicker();
852.      ImageView picture;
853.      byte[] photobytes;
854.      //save
855.      EditText thingname;
```

```
856.     MyDatabase myDatabase;
857.
858.     @Override
859.     public View onCreateView(LayoutInflater inflater, ViewGroup
   container,
860.                              Bundle savedInstanceState) {
861.         View view=inflater.inflate(R.layout.fragment_addthing,
   container, false);
862.
863.         // 自动完成文本框
864.         thingid = view.findViewById(R.id.thingid);
865.         ArrayAdapter<String> adapterauto = new
   ArrayAdapter<String>(getActivity(),
866.                 R.layout.list_item, thingids);
867.         thingid.setAdapter(adapterauto);
868.
869.         // 退出按钮
870.         Button exitButton = (Button) view.findViewById(R.id.btn_delete);
871.         exitButton.setOnClickListener(new View.OnClickListener() {
872.
873.             @Override
874.             public void onClick(View v) {
875.                 // TODO Auto-generated method stub
876.                 onCreateDialog(EXIT_DIALOG_ID);
877.
878.             }
879.         });
880.         // 出版日期
881.         final Calendar c = Calendar.getInstance();
882.         mYear = c.get(Calendar.YEAR);
883.         mMonth = c.get(Calendar.MONTH);
884.         mDay = c.get(Calendar.DAY_OF_MONTH);
885.         publishtime = (EditText) view.findViewById(R.id.publishtime);
886.         publishtime.setOnClickListener(new View.OnClickListener() {
887.
888.             @Override
889.             public void onClick(View v) {
890.                 // TODO Auto-generated method stub
891.                 onCreateDialog(DATE_DIALOG_ID);
892.
893.             }
894.         });
895.         // picture 功能
896.         picture =view.findViewById(R.id.photo);
```

```
897.         picture.setOnClickListener(new View.OnClickListener() {
898.             @Override
899.             public void onClick(View v) {
900.                 startCameraOrGallery();
901.             }
902.         });
903.
904.         //save(thingid,publishtime,picture 已有)
905.         thingname =view.findViewById(R.id.thingname);
906.         Button save=view.findViewById(R.id.btn_add);
907.         save.setOnClickListener(new View.OnClickListener() {
908.             @Override
909.             public void onClick(View v) {
910.                 String txtthingid = thingid.getText().toString();
911.                 String txtthingname = thingname.getText().toString();
912.                 String txtpublishtime =
    publishtime.getText().toString();
913.
914.                 //用 Room 数据写进数据库,默认不允许在主线程操作数据库
915.
916.                 new
    InsertThingTask(txtthingid,txtthingname,txtpublishtime,photobytes).exe
    cute();
917.
918.                 NavController
    navController=Navigation.findNavController(getActivity(),
    R.id.nav_host_fragment);
919.                 navController.navigate(R.id.nav_home);
920.             }
921.         });
922.         return view;
923.     }
924.
925.
926.
927.     // 对话框创建
928.     protected Dialog onCreateDialog(int id) {
929.         switch (id) {
930.             case EXIT_DIALOG_ID:// 退出对话框
931.                 AlertDialog.Builder builder = new AlertDialog.Builder(
932.                         getActivity());
933.                 builder.setIcon(R.drawable.ic_menu_gallery);
934.                 builder.setTitle("你确定要离开吗？");
935.                 builder.setPositiveButton("确定",
```

```
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton) {
                                // 这里添加点击确定后的逻辑
                                getActivity().finish();
                            }
                        });
                builder.setNegativeButton("取消",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,
                                                int whichButton) {
                                // 这里添加点击取消后的逻辑
                                dialog.dismiss();
                            }
                        });
                builder.create().show();
                break;
            case DATE_DIALOG_ID:// 出版日期对话框
                new DatePickerDialog(getActivity(), mDateSetListener,
    mYear, mMonth,
                        mDay).show();
                break;

        }
        return null;
    }

    // 出版日期
    private DatePickerDialog.OnDateSetListener mDateSetListener = new
    DatePickerDialog.OnDateSetListener() {

        public void onDateSet(DatePicker view, int year, int monthOfYear,
                              int dayOfMonth) {
            mYear = year;
            mMonth = monthOfYear;
            mDay = dayOfMonth;
            updateDisplay();
        }
    };

    // 出版日期 updates the date in the TextView
    private void updateDisplay() {
        publishtime.setText(new StringBuilder()
                // Month is 0 based so add 1
```

```
978.                .append(mYear).append("-").append(mMonth +
   1).append("-")
979.                .append(mDay));
980.     }
981.
982.     //picture
983.     @Override public void onActivityResult(int requestCode, int
   resultCode, Intent data) {
984.         super.onActivityResult(requestCode, resultCode, data);
985.         imagePicker.onActivityResult(this, requestCode, resultCode,
   data);
986.     }
987.
988.     @Override public void onRequestPermissionsResult(int requestCode,
   @NonNull String[] permissions,
989.                                                      @NonNull int[]
   grantResults) {
990.         super.onRequestPermissionsResult(requestCode, permissions,
   grantResults);
991.         imagePicker.onRequestPermissionsResult(this, requestCode,
   permissions, grantResults);
992.     }
993.     private void startCameraOrGallery() {
994.         new AlertDialog.Builder(getActivity()).setTitle("设置图片")
995.                .setItems(new String[] { "从相册中选取图片", "拍照" }, new
   DialogInterface.OnClickListener() {
996.                    @Override public void onClick(DialogInterface dialog,
   int which) {
997.                        // 回调
998.                        ImagePicker.Callback callback = new
   ImagePicker.Callback() {
999.                            @Override public void onPickImage(Uri imageUri)
   {
1000.                           }
1001.
1002.                           @Override public void onCropImage(Uri imageUri)
   {
1003.                               //picture.setImageURI(imageUri);
1004.                               Glide.with(getActivity()).load(new
   File(imageUri.getPath())).into(picture);
1005.                               Glide.with(getActivity()).load(new
   File(imageUri.getPath())).asBitmap().into(new SimpleTarget<Bitmap>(100,
   100) {
1006.                                  @Override
```

```
                                public void onResourceReady(Bitmap
   bitmap, GlideAnimation<? super Bitmap> glideAnimation) {
                                    ByteArrayOutputStream stream = new
   ByteArrayOutputStream();
   bitmap.compress(Bitmap.CompressFormat.JPEG, 75, stream);
                                    //savedb
                                    photobytes = stream.toByteArray();
                                }
                            });

                        }
                    };
                    if (which == 0) {
                        // 从相册中选取图片
   imagePicker.startGallery(addthingFragment.this, callback);
                    } else {
                        // 拍照
                        imagePicker.startCamera(addthingFragment.this,
   callback);
                    }
                }
            })
            .show()
            .getWindow()
            .setGravity(Gravity.CENTER);
    }

    //定义向表添加数据的的异步任务类
    public class InsertThingTask extends AsyncTask<Void,Void,Void>{
        String thingid;
        String thingname;
        String publishtime;
        byte[] img ;
        public InsertThingTask(String thingid,String thingname,String
   publishtime,byte[] img){
            this.thingid=thingid;
            this.thingname=thingname;
            this.publishtime=publishtime;
            this.img=img;
        }


```

```
1045.        @Override
1046.        protected Void doInBackground(Void... arg0) {
1047.
    MyDatabase.getInstance(getContext()).thingsDao().insertThing(new
    Thing(thingid,thingname,publishtime,img));
1048.            return null;
1049.        }
1050.        @Override
1051.        public void onPostExecute(Void result){
1052.            super.onPostExecute(result);
1053.        }
1054.
1055.    }
1056.
1057.
1058.
1059. }
1060.    @Override
1061.    public View onCreateView(LayoutInflater inflater, ViewGroup
    container,
1062.                             Bundle savedInstanceState) {
1063.        View view = inflater.inflate(R.layout.fragment_editbook,
    container, false);
1064.        txtthingid = getArguments().getString("thingid");
1065.        // 自动完成文本框
1066.        autoCompleteTextViewthingid = view.findViewById(R.id.thingid);
1067.        ArrayAdapter<String> adapterauto = new
    ArrayAdapter<String>(getActivity(),
1068.                R.layout.list_item, thingids);
1069.        autoCompleteTextViewthingid.setAdapter(adapterauto);
1070.
1071.        // 退出按钮
1072.        Button exitButton = view.findViewById(R.id.btn_delete);
1073.        exitButton.setOnClickListener(new View.OnClickListener() {
1074.
1075.            @Override
1076.            public void onClick(View v) {
1077.                // TODO Auto-generated method stub
1078.                onCreateDialog(EXIT_DIALOG_ID);
1079.
1080.            }
1081.        });
1082.        // 出版日期
1083.        final Calendar c = Calendar.getInstance();
```

```
        mYear = c.get(Calendar.YEAR);
        mMonth = c.get(Calendar.MONTH);
        mDay = c.get(Calendar.DAY_OF_MONTH);
        publishtime = (EditText) view.findViewById(R.id.publishtime);
        publishtime.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                onCreateDialog(DATE_DIALOG_ID);
            }
        });

        // picture 功能
        picture = view.findViewById(R.id.photo);
        picture.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startCameraOrGallery();
            }
        });

        //save(thingid,publishtime,picture 已有)
        thingname = view.findViewById(R.id.thingname);
        //显示数据
        showdata();

        Button save = view.findViewById(R.id.btn_add);
        save.setOnClickListener(v -> {
            String txtthingid =
    autoCompleteTextViewthingid.getText().toString();
            String txtthingname = thingname.getText().toString();
            String txtpublishertime = publishtime.getText().toString();

            //用修改后数据,更新数据库
            new UpdateStudentTask(txtthingid, txtthingname,
    txtpublishertime,photobytes).execute();


            NavController navController =
    Navigation.findNavController(getActivity(), R.id.nav_host_fragment);
            navController.navigate(R.id.nav_home);
        });
        return view;
```

```
1125.     }
1126.
1127.     private void showdata() {
1128.         Bundle bundle = getArguments();
1129.         String thingid = bundle.getString("thingid");
1130.         //查询数据库获得数据，展示在页面对应的 UI 组件
1131.         new SelectThingTask(thingid).execute();
1132.
1133.         if (photobytes != null && photobytes.length > 0) {
1134.             Bitmap image = BitmapFactory.decodeByteArray(photobytes, 0,
    photobytes.length);
1135.             picture.setImageBitmap(image);
1136.         } else {
1137.             picture.setImageResource(R.mipmap.ic_launcher);
1138.         }
1139.     }
1140. //查询数据的异步任务
1141.     private class SelectThingTask extends AsyncTask<Void, Void, Void>
1142.     {
1143.         String thingid;
1144.         Thing thing;
1145.         public SelectThingTask(String thingid)
1146.         {
1147.             this.thingid=thingid ;
1148.         }
1149.         //不能在这里更新 UI,否则有异常
1150.         @Override
1151.         protected Void doInBackground(Void... arg0)
1152.         {
1153.             thing=
    MyDatabase.getInstance(getContext()).thingsDao().getThingByThingId(thi
    ngid);
1154.             return null;
1155.         }
1156.         //onPostExecute 用于 doInBackground 执行完后，可以更新界面 UI。
1157.         @Override
1158.         protected void onPostExecute(Void result)
1159.         {
1160.             super.onPostExecute(result);
1161.             autoCompleteTextViewthingid.setText(thingid);
1162.             thingname.setText(thing.thingname);
1163.             publishtime.setText(thing.publishtime);
1164.             photobytes = thing.img;
1165.         }
```

```
1166.     }
1167.
1168.
1169.     // 更新数据库的异步任务
1170.     private class UpdateStudentTask extends AsyncTask<Void, Void, Void>
1171.     {
1172.         String thingid;
1173.         String thingname;
1174.         String publishtime ;
1175.         byte[] photobytes;
1176.
1177.         public UpdateStudentTask(String thingid, String thingname,String
     publishtime,byte[] photobytes)
1178.         {
1179.             this.thingid = thingid;
1180.             this.thingname = thingname;
1181.             this.publishtime = publishtime;
1182.             this.photobytes=photobytes;
1183.         }
1184.         @Override
1185.         protected Void doInBackground(Void... arg0)
1186.         {
1187.
     MyDatabase.getInstance(getContext()).thingsDao().updateThing(new
     Thing(thingid,thingname,publishtime,photobytes));
1188.             return null;
1189.         }
1190.
1191.         @Override
1192.         protected void onPostExecute(Void result)
1193.         {
1194.             super.onPostExecute(result);
1195.         }
1196.     }
1197. package com.example.navigationview.ui.BillList;
1198.
1199. import androidx.lifecycle.ViewModelProvider;
1200.
1201. import android.annotation.SuppressLint;
1202. import android.content.Intent;
1203. import android.database.Cursor;
1204. import android.database.sqlite.SQLiteDatabase;
1205. import android.graphics.Bitmap;
1206. import android.os.Bundle;
```

```
1207.
1208. import androidx.annotation.NonNull;
1209. import androidx.annotation.Nullable;
1210. import androidx.fragment.app.Fragment;
1211. import androidx.recyclerview.widget.LinearLayoutManager;
1212. import androidx.recyclerview.widget.RecyclerView;
1213.
1214. import android.view.LayoutInflater;
1215. import android.view.View;
1216. import android.view.ViewGroup;
1217. import android.widget.Button;
1218. import android.widget.TextView;
1219.
1220. import com.example.navigationview.ManageActivity;
1221. import com.example.navigationview.R;
1222. import com.example.navigationview.database.DBHelper;
1223.
1224. import java.util.ArrayList;
1225. import java.util.HashMap;
1226. import java.util.List;
1227. import java.util.Map;
1228.
1229. public class BillListFragment extends Fragment {
1230.
1231.
1232.     private BillListViewModel mViewModel;
1233.
1234.     RecyclerView recyclerView;
1235.     List<Map<String, Object>> mData;
1236.
1237.     private static final String DATABASE_NAME = "MyThings.db";
1238.     private static final String TABLE_NAME = "record";
1239.     private static final String COLUMN_ID = "id";
1240.     private static final String COLUMN_DATE = "date";
1241.     private static final String COLUMN_TYPE = "type";
1242.     private static final String COLUMN_MONEY = "money";
1243.     private static final String COLUMN_STATE = "state";
1244.
1245.
1246.     public static BillListFragment newInstance() {
1247.         return new BillListFragment();
1248.     }
1249.
1250.     @Override
```

```
1251.    public View onCreateView(@NonNull LayoutInflater inflater,
   @Nullable ViewGroup container,
1252.                            @Nullable Bundle savedInstanceState) {
1253.        View view=inflater.inflate(R.layout.fragment_bill_list,
   container, false);
1254.
1255.        // 添加按钮
1256.        Button addButton = view.findViewById(R.id.addbutton);
1257.        addButton.setOnClickListener(new View.OnClickListener() {
1258.
1259.            @Override
1260.            public void onClick(View v) {
1261.                // TODO Auto-generated method stub
1262.                Intent intent1=new Intent(getActivity(),
   ManageActivity.class);
1263.                startActivity(intent1);
1264.
1265.            }
1266.        });
1267.
1268.        recyclerView=view.findViewById(R.id.recycler_view);
1269.        mData=getData();
1270.        recyclerView.setLayoutManager(new
   LinearLayoutManager(getActivity()));//垂直线性布局
1271.        //recyclerView.setLayoutManager(new
   GridLayoutManager(getActivity(),2));//线性宫格显示，类似 gridview
1272.        // recyclerView.setLayoutManager(new
   StaggeredGridLayoutManager(3, OrientationHelper.VERTICAL));//线性宫格显示
   类似瀑布流
1273.        recyclerView.setAdapter(new
   BillListFragment.MyRecycleViewAdapter());
1274.
1275.        return view;
1276.
1277.    }
1278.
1279.    @SuppressLint("Range")
1280.    private List<Map<String, Object>> getData() {
1281.
1282.        List<Map<String, Object>> list = new ArrayList<Map<String,
   Object>>();
1283.        byte[] b = null;
1284.        Bitmap image = null;
1285.        //查询数据库获得数据
```

```
1286.        DBHelper dbOpenHelper =new DBHelper((getActivity()));
1287.        SQLiteDatabase db=dbOpenHelper.getWritableDatabase();
1288.        Cursor cursor =db.rawQuery("select * from record",null);
1289.        //把查询数据封装到 Cursor
1290. //       Cursor cursor = sqLiteDatabase.rawQuery("select * from record",
     null);
1291. //       ArrayList<Map<String, String>> list = new ArrayList<Map<String,
     String>>();
1292.        //用 while 循环遍历 Cursor，再把它分别放到 map 中，最后统一存入 list 中，
     便于调用
1293.        while (cursor.moveToNext()) {
1294.
1295.            int id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID));
1296.            String date =
     cursor.getString(cursor.getColumnIndex(COLUMN_DATE));
1297.            String type =
     cursor.getString(cursor.getColumnIndex(COLUMN_TYPE));
1298.            float money =
     cursor.getFloat(cursor.getColumnIndex(COLUMN_MONEY));
1299.            String state =
     cursor.getString(cursor.getColumnIndex(COLUMN_STATE));
1300.            HashMap<String,Object> map=new HashMap<>();
1301. //           Map<String, String> map = new HashMap<String, String>();
1302.            map.put("id", String.valueOf(id));
1303.            map.put("date", date);
1304.            map.put("type", type);
1305.            map.put("money", String.valueOf(money));
1306.            map.put("state", state);
1307.            list.add(map);
1308.        }
1309.        cursor.close();
1310.        db.close();
1311.
1312.        return list;
1313.    }
1314.    class  MyRecycleViewAdapter extends
     RecyclerView.Adapter<BillListFragment.MyRecycleViewAdapter.ViewHolder>
1315.    {
1316.
1317.
1318.        public  class ViewHolder extends RecyclerView.ViewHolder {
1319.            public TextView id;
1320.            public TextView date;
1321.            public TextView type;
```

```
1322.           public TextView money;
1323.           public TextView state;
1324.
1325.           public ViewHolder(View convertView) {
1326.               super(convertView);
1327.               id= (TextView)convertView.findViewById(R.id.list_id);
1328.               date=
    (TextView)convertView.findViewById(R.id.list_date);
1329.               type=
    (TextView)convertView.findViewById(R.id.list_type);
1330.               money=
    (TextView)convertView.findViewById(R.id.list_money);
1331.               state =
    (TextView)convertView.findViewById(R.id.list_state);
1332.           }
1333.       }
1334.       @NonNull
1335.       @Override
1336.       public BillListFragment.MyRecycleViewAdapter.ViewHolder
    onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
1337.           View v=
    LayoutInflater.from(getActivity()).inflate(R.layout.record_item_layout,
    parent, false);
1338.           final RecyclerView
    recyclerView=v.findViewById(R.id.recycler_view);
1339.
1340.           return new
    BillListFragment.MyRecycleViewAdapter.ViewHolder(v);
1341.       }
1342.       @Override
1343.       public void onBindViewHolder(@NonNull
    BillListFragment.MyRecycleViewAdapter.ViewHolder holder,
    @SuppressLint("RecyclerView") final int position) {
1344.
1345.           holder.id.setText((String)mData.get(position).get("id"));
1346.
    holder.date.setText((String)mData.get(position).get("date"));
1347.
    holder.type.setText((String)mData.get(position).get("type"));
1348.
    holder.money.setText((String)mData.get(position).get("money"));
1349.
    holder.state.setText((String)mData.get(position).get("state"));
```

```
1350. //          holder.itemView.setOnClickListener(new
   View.OnClickListener() {
1351. //              @Override
1352. //              public void onClick(View v) {
1353. //                  NavController navController=
   Navigation.findNavController(getActivity(),R.id.nav_host_fragment);
1354. //                  Bundle bundle=new Bundle();
1355. //
   bundle.putString("id",(String)mData.get(position).get("id"));
1356. //
   navController.navigate(R.id.detailFragment,bundle);
1357. //
1358. //              }
1359. //          });
1360.         }
1361.
1362.         @Override
1363.         public int getItemCount() {
1364.             return mData.size();
1365.         }
1366.     }
1367.
1368.     @Override
1369.     public void onActivityCreated(@Nullable Bundle savedInstanceState)
   {
1370.         super.onActivityCreated(savedInstanceState);
1371.         mViewModel = new
   ViewModelProvider(this).get(BillListViewModel.class);
1372.         // TODO: Use the ViewModel
1373.     }
1374.
1375.
1376. }
1377. package com.example.navigationview.ui.mypage;
1378.
1379. import androidx.appcompat.app.AlertDialog;
1380. import androidx.lifecycle.ViewModelProvider;
1381.
1382. import android.content.DialogInterface;
1383. import android.content.Intent;
1384. import android.os.Bundle;
1385.
1386. import androidx.annotation.NonNull;
1387. import androidx.annotation.Nullable;
```

```
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import com.example.navigationview.LoginActivity;
import com.example.navigationview.R;
import com.example.navigationview.User;

import java.util.ArrayList;

public class MyPageFragment extends Fragment {



    private MyPageViewModel mViewModel;

    public static MyPageFragment newInstance() {
        return new MyPageFragment();
    }
    ArrayList<User> list;
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
   @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        View view=inflater.inflate(R.layout.fragment_my_page,
   container, false);
        Intent intent = getActivity().getIntent();

        list =intent.getParcelableArrayListExtra("LoginUser");
        User user=list.get(0);
        final String username=user.getUserId();

        TextView tv_welcome=view.findViewById(R.id.tv_welcome);
        tv_welcome.setText("欢迎您 ,用户"+username);

        //退出按键
        ImageView btn_exit=view.findViewById(R.id.btn_exit);
        btn_exit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
1430.              AlertDialog dialog = new
   AlertDialog.Builder(getActivity()).setTitle("退出操作")
1431.                      .setMessage("确定退出，不继续玩玩？")
1432.                      .setPositiveButton("确定", new
   DialogInterface.OnClickListener() {
1433.                          @Override
1434.                          public void onClick(DialogInterface dialog, int
   which) {
1435.                              Intent intent=new Intent(getActivity(),
   LoginActivity.class);
1436.                              startActivity(intent);
1437.                          }
1438.                      }).setNegativeButton("继续留下！", new
   DialogInterface.OnClickListener() {
1439.                          @Override
1440.                          public void onClick(DialogInterface dialog, int
   which) {
1441.
1442.                          }
1443.                      }).create();
1444.              dialog.show();
1445.
1446.          }
1447.      });
1448.
1449.
1450.      return view;
1451.  }
1452.
1453.  @Override
1454.  public void onActivityCreated(@Nullable Bundle savedInstanceState)
   {
1455.      super.onActivityCreated(savedInstanceState);
1456.      mViewModel = new
   ViewModelProvider(this).get(MyPageViewModel.class);
1457.      // TODO: Use the ViewModel
1458.  }
1459.
1460. }
1461. package com.example.navigationview.ui.recyclerview;
1462.
1463. import android.content.DialogInterface;
1464. import android.content.Intent;
1465. import android.os.Bundle;
```

```
1466.
1467. import androidx.annotation.NonNull;
1468. import androidx.annotation.Nullable;
1469. import androidx.appcompat.app.AlertDialog;
1470. import androidx.fragment.app.Fragment;
1471. import androidx.lifecycle.Observer;
1472. import androidx.lifecycle.ViewModelProvider;
1473. import androidx.navigation.NavController;
1474. import androidx.navigation.Navigation;
1475. import androidx.recyclerview.widget.OrientationHelper;
1476. import androidx.recyclerview.widget.RecyclerView;
1477. import androidx.recyclerview.widget.StaggeredGridLayoutManager;
1478.
1479. import android.view.LayoutInflater;
1480. import android.view.View;
1481. import android.view.ViewGroup;
1482. import android.widget.Button;
1483. import android.widget.ImageView;
1484. import android.widget.TextView;
1485.
1486. import com.example.navigationview.LoginActivity;
1487. import com.example.navigationview.ManageActivity;
1488. import com.example.navigationview.R;
1489. import com.example.navigationview.SearchRecordActivity;
1490. import com.example.navigationview.database.Thing;
1491.
1492. import java.util.ArrayList;
1493. import java.util.List;
1494. import java.util.Map;
1495.
1496.
1497. public class RecyclerViewFragment extends Fragment {
1498.
1499.     private RecyclerViewViewModel mViewModel;
1500.     private RecyclerView recyclerView;
1501.     private List<Thing> thingList;
1502.     List<Map<String,Object>> mydata;
1503.     MyRecycleViewAdapter myRecycleViewAdapter;
1504.
1505.     public static RecyclerViewFragment newInstance() {
1506.         return new RecyclerViewFragment();
1507.     }
1508.
1509.     @Override
```

```
1510.    public View onCreateView(@NonNull LayoutInflater inflater,
   @Nullable ViewGroup container,
1511.                             @Nullable Bundle savedInstanceState) {
1512.           View view = inflater.inflate(R.layout.fragment_recyclerview,
   container, false);
1513.        recyclerView = view.findViewById(R.id.recycler_view);
1514.        //recyclerView.setLayoutManager(new
   LinearLayoutManager(getActivity()));//垂直线性布局
1515. //       recyclerView.setLayoutManager(new
   GridLayoutManager(getActivity(),2));//线性宫格显示，类似 gridview
1516.        recyclerView.setLayoutManager(new StaggeredGridLayoutManager(3,
   OrientationHelper.VERTICAL));//线性宫格显示类似瀑布流
1517.        thingList = new ArrayList<>();
1518.        myRecycleViewAdapter = new
   MyRecycleViewAdapter(getContext(),thingList);
1519.        recyclerView.setAdapter(myRecycleViewAdapter);
1520.
1521.        //从 RecyclerViewViewModel 获取数据，观察 livedata
1522.        RecyclerViewViewModel recyclerViewViewModel = new
   ViewModelProvider(this).get(RecyclerViewViewModel.class);
1523.
   recyclerViewViewModel.getLiveDataThing().observe(getViewLifecycleOwner
   (), new Observer<List<Thing>>() {
1524.            @Override
1525.            public void onChanged(List<Thing> things) {
1526.                thingList.clear();
1527.                thingList.addAll(things);
1528.                myRecycleViewAdapter.notifyDataSetChanged();
1529.            }
1530.        });
1531.
1532.
1533.        return view;
1534.    }
1535.
1536.
1537.    // 定义一个 RecyclerView 专属的数据适配器
1538.    class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder>
1539.    {
1540.        @NonNull
1541.        @Override
1542.        public MyAdapter.ViewHolder onCreateViewHolder(@NonNull
   ViewGroup parent, int viewType) {
```

```
1543.           View view=
   LayoutInflater.from(getContext()).inflate(R.layout.item,parent,false);
1544.           ViewHolder viewHolder=new ViewHolder(view);
1545.
1546.           Button addButton = view.findViewById(R.id.addbutton);
1547.           addButton.setOnClickListener(new View.OnClickListener() {
1548.               @Override
1549.               public void onClick(View v) {
1550.                   // TODO Auto-generated method stub
1551.                   Intent intent1=new Intent(getActivity(),
   ManageActivity.class);
1552.                   startActivity(intent1);
1553.
1554.               }
1555.           });
1556.
1557.           Button searchButton = view.findViewById(R.id.searchbutton);
1558.           searchButton.setOnClickListener(new View.OnClickListener()
   {
1559.               @Override
1560.               public void onClick(View v) {
1561.                   // TODO Auto-generated method stub
1562.                   Intent intent1=new Intent(getActivity(),
   SearchRecordActivity.class);
1563.                   startActivity(intent1);
1564.               }
1565.           });
1566.
1567.           Button addthingButton =
   view.findViewById(R.id.addthingbutton);
1568.           addthingButton.setOnClickListener(new
   View.OnClickListener() {
1569.               @Override
1570.               public void onClick(View v) {
1571.                   // TODO Auto-generated method stub
1572.                   Intent intent1=new Intent(getActivity(),
   SearchRecordActivity.class);
1573.                   startActivity(intent1);
1574.               }
1575.           });
1576.
1577.           //退出按键
1578.           Button btn_exit=view.findViewById(R.id.btn_exit);
1579.           btn_exit.setOnClickListener(new View.OnClickListener() {
```

```
1580.              @Override
1581.              public void onClick(View v) {
1582.                  AlertDialog dialog = new
   AlertDialog.Builder(getActivity()).setTitle("退出操作")
1583.                          .setMessage("确定退出，不继续玩玩？")
1584.                          .setPositiveButton("确定", new
   DialogInterface.OnClickListener() {
1585.                              @Override
1586.                              public void onClick(DialogInterface dialog,
   int which) {
1587.                                  Intent intent=new Intent(getActivity(),
   LoginActivity.class);
1588.                                  startActivity(intent);
1589.                              }
1590.                          }).setNegativeButton("继续留下！", new
   DialogInterface.OnClickListener() {
1591.                              @Override
1592.                              public void onClick(DialogInterface dialog,
   int which) {
1593.
1594.                              }
1595.                          }).create();
1596.                  dialog.show();
1597.
1598.              }
1599.          });
1600.
1601.          return viewHolder;
1602.      }
1603.
1604.      @Override
1605.      public void onBindViewHolder(@NonNull MyAdapter.ViewHolder
   holder, int position) {
1606.          final String
   title=(String)mydata.get(position).get("title");
1607.          final String info= (String)
   mydata.get(position).get("info");
1608.          final int img= (Integer) mydata.get(position).get("img");
1609.          holder.title.setText(title);
1610.          holder.info.setText(info);
1611.          holder.img.setBackgroundResource(img);
1612.          //新增代码
1613.          holder.itemView.setOnClickListener(new
   View.OnClickListener() {
```

```
1614.              @Override
1615.              public void onClick(View v) {
1616.                  NavController navController=
   Navigation.findNavController(getActivity(),R.id.thingFragment);
1617.                  Bundle bundle=new Bundle();
1618.                  bundle.putString("title",title);
1619.                  bundle.putString("info",info);
1620.                  bundle.putInt("img",img);
1621.
   navController.navigate(R.id.action_detailFragment_to_showbookthingFrag
   ment,bundle);
1622.              }
1623.          });
1624.
1625.
1626.      }
1627.
1628.      @Override
1629.      public int getItemCount() {
1630.          return mydata.size();
1631.      }
1632.      public class ViewHolder extends RecyclerView.ViewHolder {
1633.          TextView title,info;
1634.          ImageView img;
1635.
1636.          public ViewHolder(@NonNull View itemView) {
1637.              super(itemView);
1638.              title=itemView.findViewById(R.id.title);
1639.              info=itemView.findViewById(R.id.info);
1640.              img=itemView.findViewById(R.id.img);
1641.
1642.          }
1643.      }
1644.  }
1645.
1646. }
1647. package com.example.navigationview.ui.QueryBill;
1648.
1649. import androidx.lifecycle.ViewModelProvider;
1650.
1651. import android.annotation.SuppressLint;
1652. import android.database.Cursor;
1653. import android.database.sqlite.SQLiteDatabase;
1654.
```

```
1655. import android.graphics.Bitmap;
1656. import android.graphics.BitmapFactory;
1657. import android.os.Bundle;
1658.
1659. import androidx.annotation.NonNull;
1660. import androidx.annotation.Nullable;
1661. import androidx.fragment.app.Fragment;
1662. import androidx.navigation.NavController;
1663. import androidx.navigation.Navigation;
1664. import androidx.recyclerview.widget.GridLayoutManager;
1665. import androidx.recyclerview.widget.LinearLayoutManager;
1666. import androidx.recyclerview.widget.RecyclerView;
1667.
1668. import android.view.LayoutInflater;
1669. import android.view.View;
1670. import android.view.ViewGroup;
1671. import android.widget.AdapterView;
1672. import android.widget.EditText;
1673. import android.widget.ImageView;
1674. import android.widget.ListView;
1675. import android.widget.SimpleAdapter;
1676. import android.widget.Spinner;
1677. import android.widget.TextView;
1678.
1679. import com.example.navigationview.R;
1680. import com.example.navigationview.database.DBHelper;
1681.
1682. import java.util.ArrayList;
1683. import java.util.HashMap;
1684. import java.util.List;
1685. import java.util.Map;
1686.
1687. public class QueryBillFragment extends Fragment {
1688.
1689.     private QueryBillViewModel mViewModel;
1690.     RecyclerView recyclerView;
1691.     List<Map<String, Object>> mData;
1692.
1693.
1694.     //定义 spinner 中的数据
1695.     private String[] date_data= {"", "202201", "202202", "202203",
    "202204",
    "202205","202206","202207","202208","202209","202210","202211","202212
    "};
```

```
1696.    private String[] type_data = {"", "收入", "支出"};
1697.    Spinner spin_date, spin_type;
1698.    ListView listView;
1699.    TextView tv_show;
1700.    float sum=0;
1701.
1702.    //数据库
1703.    private String selectDate, selectType;
1704.    private static final String DATABASE_NAME = "MyThings.db";
1705.    private static final String TABLE_NAME = "record";
1706.    private static final String COLUMN_ID = "id";
1707.    private static final String COLUMN_DATE = "date";
1708.    private static final String COLUMN_TYPE = "type";
1709.    private static final String COLUMN_MONEY = "money";
1710.    private static final String COLUMN_STATE = "state";
1711.    private SQLiteDatabase sqLiteDatabase = null;
1712.
1713.
1714.    public static QueryBillFragment newInstance() {
1715.        return new QueryBillFragment();
1716.    }
1717.
1718.    @Override
1719.    public View onCreateView(@NonNull LayoutInflater inflater,
   @Nullable ViewGroup container,
1720.                             @Nullable Bundle savedInstanceState) {
1721.
1722.        View view= inflater.inflate(R.layout.fragment_query_bill,
   container, false);
1723.        recyclerView=view.findViewById(R.id.recycler_view);
1724.        mData=getData();
1725.        recyclerView.setLayoutManager(new
   LinearLayoutManager(getActivity()));//垂直线性布局
1726.        //recyclerView.setLayoutManager(new
   GridLayoutManager(getActivity(),2));//线性宫格显示，类似 gridview
1727.        // recyclerView.setLayoutManager(new
   StaggeredGridLayoutManager(3, OrientationHelper.VERTICAL));//线性宫格显示
   类似瀑布流
1728.        recyclerView.setAdapter(new MyRecycleViewAdapter());
1729.        return view;
1730.    }
1731.
1732.
1733.    @SuppressLint("Range")
```

```
1734.    private List<Map<String, Object>> getData() {
1735.
1736.        List<Map<String, Object>> list = new ArrayList<Map<String,
   Object>>();
1737.        byte[] b = null;
1738.        Bitmap image = null;
1739.        //查询数据库获得数据
1740.        DBHelper dbOpenHelper =new DBHelper((getActivity()));
1741.        SQLiteDatabase db=dbOpenHelper.getWritableDatabase();
1742.        Cursor cursor =db.rawQuery("select * from record",null);
1743.        //把查询数据封装到 Cursor
1744. //       Cursor cursor = sqLiteDatabase.rawQuery("select * from record",
   null);
1745. //       ArrayList<Map<String, String>> list = new ArrayList<Map<String,
   String>>();
1746.        //用 while 循环遍历 Cursor，再把它分别放到 map 中，最后统一存入 list 中，
   便于调用
1747.        while (cursor.moveToNext()) {
1748.
1749.            int id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID));
1750.            String date =
   cursor.getString(cursor.getColumnIndex(COLUMN_DATE));
1751.            String type =
   cursor.getString(cursor.getColumnIndex(COLUMN_TYPE));
1752.            float money =
   cursor.getFloat(cursor.getColumnIndex(COLUMN_MONEY));
1753.            String state =
   cursor.getString(cursor.getColumnIndex(COLUMN_STATE));
1754.            HashMap<String,Object>map=new HashMap<>();
1755. //           Map<String, String> map = new HashMap<String, String>();
1756.            map.put("id", String.valueOf(id));
1757.            map.put("date", date);
1758.            map.put("type", type);
1759.            map.put("money", String.valueOf(money));
1760.            map.put("state", state);
1761.            list.add(map);
1762.        }
1763.        cursor.close();
1764.        db.close();
1765.
1766.        return list;
1767.    }
1768.    class  MyRecycleViewAdapter extends
   RecyclerView.Adapter<MyRecycleViewAdapter.ViewHolder>
```

```
1769.    {
1770.
1771.
1772.        public  class ViewHolder extends RecyclerView.ViewHolder {
1773.            public TextView id;
1774.            public TextView date;
1775.            public TextView type;
1776.            public TextView money;
1777.            public TextView state;
1778.
1779.            public ViewHolder(View convertView) {
1780.                super(convertView);
1781.                id= (TextView)convertView.findViewById(R.id.list_id);
1782.                date=
    (TextView)convertView.findViewById(R.id.list_date);
1783.                type=
    (TextView)convertView.findViewById(R.id.list_type);
1784.                money=
    (TextView)convertView.findViewById(R.id.list_money);
1785.                state =
    (TextView)convertView.findViewById(R.id.list_state);
1786.            }
1787.        }
1788.        @NonNull
1789.        @Override
1790.        public MyRecycleViewAdapter.ViewHolder
    onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
1791.            View v=
    LayoutInflater.from(getActivity()).inflate(R.layout.record_item_layout,
    parent, false);
1792.            final RecyclerView
    recyclerView=v.findViewById(R.id.recycler_view);
1793.
1794.            return new ViewHolder(v);
1795.        }
1796.        @Override
1797.        public void onBindViewHolder(@NonNull
    MyRecycleViewAdapter.ViewHolder holder, @SuppressLint("RecyclerView")
    final int position) {
1798.
1799.            holder.id.setText((String)mData.get(position).get("id"));
1800.
    holder.date.setText((String)mData.get(position).get("date"));
```

```
1801.
   holder.type.setText((String)mData.get(position).get("type"));
1802.
   holder.money.setText((String)mData.get(position).get("money"));
1803.
   holder.state.setText((String)mData.get(position).get("state"));
1804. //          holder.itemView.setOnClickListener(new
   View.OnClickListener() {
1805. //              @Override
1806. //              public void onClick(View v) {
1807. //                  NavController navController=
   Navigation.findNavController(getActivity(),R.id.nav_host_fragment);
1808. //                  Bundle bundle=new Bundle();
1809. //
   bundle.putString("id",(String)mData.get(position).get("id"));
1810. //
   navController.navigate(R.id.detailFragment,bundle);
1811. //
1812. //              }
1813. //          });
1814.       }
1815.
1816.       @Override
1817.       public int getItemCount() {
1818.           return mData.size();
1819.       }
1820.    }
1821.
1822.    @Override
1823.    public void onActivityCreated(@Nullable Bundle savedInstanceState)
   {
1824.       super.onActivityCreated(savedInstanceState);
1825.       mViewModel = new
   ViewModelProvider(this).get(QueryBillViewModel.class);
1826.       // TODO: Use the ViewModel
1827.    }
1828.
1829.
1830.
1831.
1832. }
1833.    @Override
1834.    public View onCreateView(@NonNull LayoutInflater inflater,
   @Nullable ViewGroup container,
```

```
                        @Nullable Bundle savedInstanceState) {
        View view=inflater.inflate(R.layout.fragment_viewpages2,
    container, false);
        view_banner=view.findViewById(R.id.bannerVp);

    layout_dot=view.findViewById(R.id.ViewpagerFragment_view_dot);
        imgUrl=new ViewpagerModel().getData();
        bannerAdapter=new BannerAdapter(getContext(),imgUrl);
        view_banner.setAdapter(bannerAdapter);
        dotList=new ArrayList<>();
        initIndicatorDots();
        view_banner.registerOnPageChangeCallback(new
    ViewPager2.OnPageChangeCallback() {
            @Override
            public void onPageSelected(int position) {
                super.onPageSelected(position);
                for(int i = 0; i <imgUrl.size(); i++){
                    if(i==position%imgUrl.size()){

    dotList.get(i).setBackgroundResource(R.drawable.shape_dot_red);
                    }else{

    dotList.get(i).setBackgroundResource(R.drawable.shape_dot_white);
                    }
                }
            }
        });
        return view;
    }

    //轮播图效果的实现
    private void initViews(View view) {
        imgUrl=new ViewpagerModel().getData();//获取轮播图的图片数据
        view_banner = view.findViewById(R.id.bannerVp);
        layout_dot=
    view.findViewById(R.id.ViewpagerFragment_view_dot);
        bannerAdapter = new BannerAdapter(getContext(),imgUrl);
        view_banner.setAdapter(bannerAdapter);
        dotList=new ArrayList<>();
        initIndicatorDots();
        //注册轮播图的滚动事件监听器
        }
    @Override
```

```
1872.    public void onActivityCreated(@Nullable Bundle savedInstanceState)
   {
1873.        super.onActivityCreated(savedInstanceState);
1874.        mViewModel = new
   ViewModelProvider(this).get(Viewpages2ViewModel.class);
1875.        // TODO: Use the ViewModel
1876.    }
```