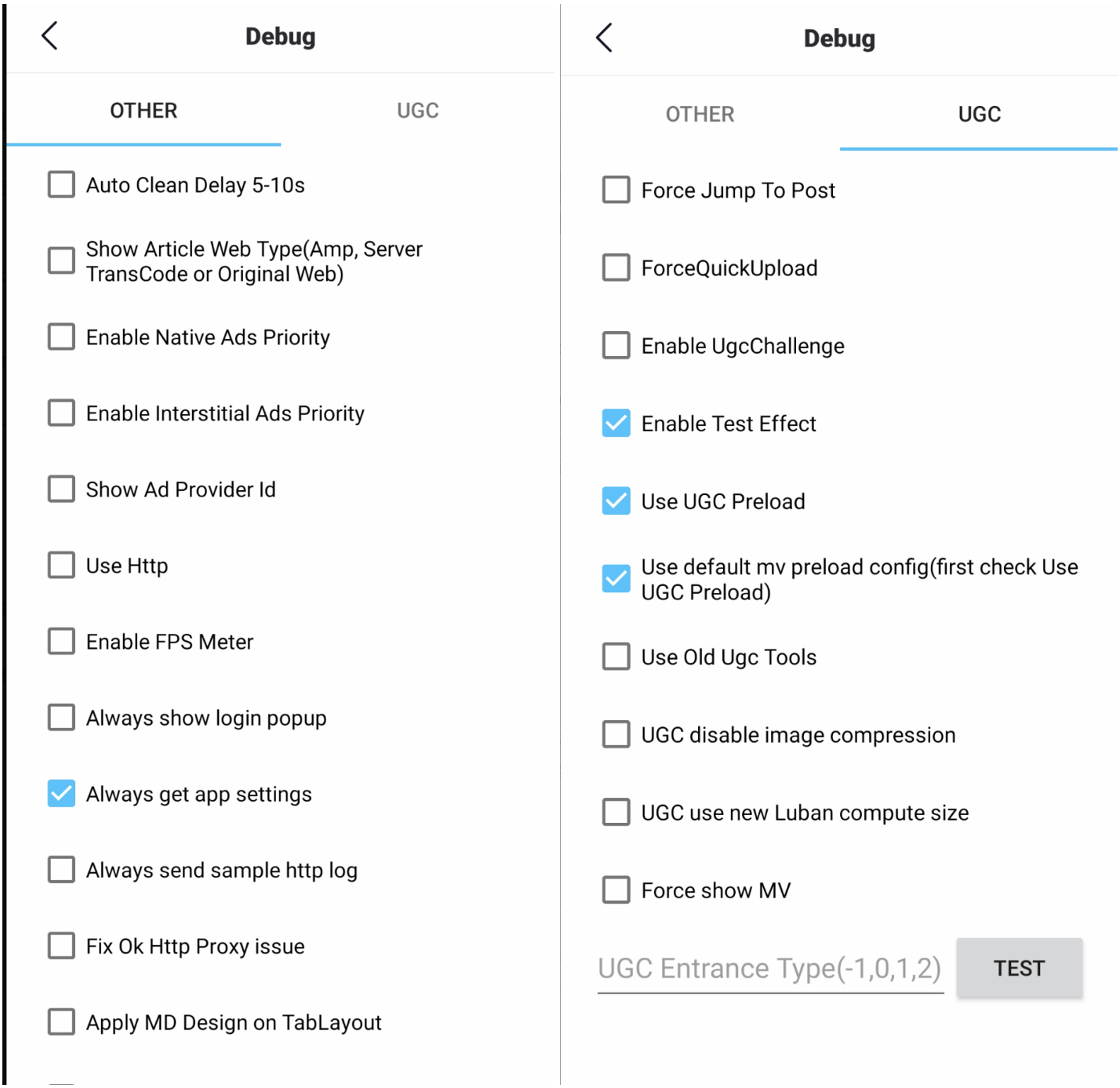
Debug模块设计

需求介绍

a. 简化开发人员添加Debug模块的操作
b. 将Debug模块按照小组进行分类



技术设计

a. 定义标准的数据类(DebugDataModel)及其所对应ViewHolder，并根据不同的控件扩展了四种不同的数据类型及ViewHolder，用以简化不同控件的操作。

```
/**
 * 存放所有Debug页面的初始数据及其类型
 */

open class DebugDataModel(val text:String,val viewType: ViewType)

class DebugCheckBoxModel(mText:String,val realization:(viewHolder:CheckBoxViewHolder)->Unit):DebugDataModel(mText,ViewType.CheckBox)
class DebugEditTextModel(mText:String,val realization:(viewHolder:EditTextViewHolder)->Unit):DebugDataModel(mText,ViewType.EditText)
class DebugTextViewModel(mText:String,val realization:(viewHolder:TextViewViewHolder)->Unit):DebugDataModel(mText,ViewType.TextView)
class DebugSpinnerModel(mText:String,val realization:(viewHolder:SpinnerViewHolder)->Unit):DebugDataModel(mText,ViewType.Spinner)
```

```
/**
 * 4种不同的类型对应4种ViewHolder
 */
abstract class ViewHolder(view: View): RecyclerView.ViewHolder(view){...}
class EditTextViewHolder(view: View) : ViewHolder(view) {...}
class CheckBoxViewHolder(view: View): ViewHolder(view){...}
class TextViewViewHolder(view: View): ViewHolder(view){...}
class SpinnerViewHolder(view: View) : ViewHolder(view) {...}
```

b. 定义ViewPager+TabLayout的主体框架，使得每一个小组对应一个Fragment，每个Fragment对应其数据集合。

c. 为所有的Fragment的布局(RecyclerView)，并为其定义标准的Adapter，Adapter根据控件的不同类型回调其绑定的方法并为开发者提供该类型数据所对应的ViewHolder类的变量holder，开发者只需在添加时操作该holder变量即可完成该控件的逻辑实现。

```
open fun bindEditTextViewHolder(holder: EditTextViewHolder,item: DebugEditTextModel){
    item.realization.invoke(holder)
}
open fun bindCheckBoxViewHolder(holder: CheckBoxViewHolder,item: DebugCheckBoxModel){
    item.realization.invoke(holder)
}
open fun bindTextViewViewHolder(holder: TextViewViewHolder,item: DebugTextViewModel){
    item.realization.invoke(holder)
}
open fun bindSpinnerViewHolder(holder: SpinnerViewHolder,item: DebugSpinnerModel){
    item.realization.invoke(holder)
}
```

使用说明

开发者只需在DebugFragment中的onCreateView()方法中调用addDepartments()方法，并传入对应的标题title及数据集合list，即可完成添加或改动当前小组所对应Fragment中的数据。