**《移动应用软件开发》期末软件设计报告**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | | **2021 级 计算机科学与技术（卓越）1班班** | | | **姓名** | **韩昊辰** | | **学号** | **20214272** |
| **提交时间** | **2023.12.10** | | **学年学期** | **2023-2024（1）** | | | **指导教师** | | **杨瑞龙** |
| **软件名称** | 快乐家务APP | | | | | | | | |
| 说明：   1. 每人独立完成移动应用软件设计开发，并完成设计报告。 2. 自拟软件题目。不准抄袭，应当是自己独立编写的软件。抄袭计0分。 3. **软件开发平台选择HarmonyOS，优先使用ArkUI开发。** 4. 从界面、技术、功能等方面对软件进行评价。程序有必要的注释。 5. 从文档条理性、文档规范性、内容完整性等方面对设计报告进行评价。 6. 所开发的程序源代码压缩成ZIP格式，命名为：学号姓名-期末软件设计.zip。报告命名为：学号姓名-期末软件设计报告.docx。演示视频清晰，MP4格式，大小不超过20M，命名为：学号姓名-期末软件设计演示.mp4。**请在谷歌或者搜狗浏览器预览视频是否能够正常播放。三个文件分别提交到云班课。** | | | | | | | | | |
| 一、软件设计意图、功能介绍和特色  **1.1 软件设计意图**  在当今快节奏的生活中，家庭中有小孩的父母往往面临繁忙的工作和家务负担。为了更好地管理家务事项，并培养小朋友的责任心和主人翁精神，我开发一款创新的快乐家务助手APP。  家务助手APP旨在智能管理家庭中的各项家务事项，为家庭成员提供更加便捷、高效的生活方式。其中，特别注重培养小朋友的家务意识。  **1.2 软件功能介绍**  软件实现了以下功能：   * 智能组织家务任务： 家务助手APP能够根据家务日期，时间，完成者，奖励等智能组织家务显示列表，支持多种排序方式，提供方便快捷的家务新建，查看和删除功能。并快速按照关键词查询家务 * 闹钟提醒： 可为每个家务任务设置闹钟提醒，以确保成员按时完成任务。这有助于培养孩子们的时间管理能力，提高工作效率。 * 完成度奖励： 家庭成员完成家务任务后，系统将根据任务的难度和完成效果给予相应的奖励。这不仅可以激发小朋友的积极性，还能让成员感受到努力工作的成果。 * 指定完成人： 家长可以根据实际情况指定特定的家庭成员完成某项家务任务，有助于合理分工，提高任务完成的效率。 * 任务鲜花： 家庭成员通过完成任务可以获得鲜花，这些鲜花可以用于参与抽奖功能。这样一来，完成家务不仅是一种责任，还成为了一项有趣的挑战。 * 抽奖功能： 家庭成员可以使用获得的鲜花参与抽奖，有机会赢取各种奖品，激发大家更积极地参与家务活动。 * 家庭成员管理：软件可管理家庭成员，根据家庭成员的身份：父母小孩，或是保姆来客赋予标签，并管理家庭成员的鲜花数目   **1.3 软件特色功能介绍**  **本APP实现了诸多亮点功能，以下是主要亮点功能和介绍：**   * 支持将【日期，家务制定完成者，排序方式】的一个或多个作为条件，以**任意形式组合查询家务** * 支持**为家务指定完成者和奖励**，并在完成时**为完成者提供奖励** * 支持**评价性完成家务**，按照完成度以五级评分**权重赋予奖励** * 支持按照【时间，奖励，只看已完成，只看未完成】三种方式**排序家务记录** * 支持**升序/降序查看家务** * 支持根据**关键词查询家务**，标题、备注、完成者中包含该关键词的家务会被查询显示 * 支持**为家务添加闹钟提醒**，在任务指定完成时间到来时刻进行，在手机任何位置（无论本APP是否打开）弹窗和发声提醒，点击弹窗后**跳转到APP首页** * 支持消耗**鲜花抽奖，**使用Carvas画布**实现抽奖转盘**   下表展示了亮点功能和实现亮点功能所用技术，展示了作者为了实现特色功能自学的成果和努力。  用到的关键技术包括：**关系型数据库——rdb，sql表连接，非关系型数据库——首选项preferences，map、record、对象型数组实现的排序操作，Reminder提醒类调用系统后台代理，Carvas画布**  *表格 1 亮点功能和实现亮点功能所用技术*   |  |  | | --- | --- | | **特色功能** | **实现技术** | | **任意条件组合查询家务** | **将Page页面用户选择的条件作为指令，根据条件判断标志判断用户是否对某条件进行选择，若选择，通过rdb提供的接口，使用and()扩展原sql查询条件，任意条件间通过and()拼接为新的sql，查询到指定数据** | | **按照【时间，奖励，只看已完成，只看未完成】三种方式排序流水记录** | **选中一个排序方式后，进行数据库查询。其实，选中【日期】或【指定家务完成者】或【关键词查询】确认后，都会进行数据库查询。查询语句根据当前页面所有选中参数，包括排序，传参给查询语句。数据库查询后，将结果在页面中进行渲染。** | | **支持升序/降序查看流水记录** | **对已排序的对象型数组进行reverse操作，同时在全局有一个记录位记录当前数组是顺序/逆序** | | **支持为家务指定完成者和奖励，并在完成时为完成者提供奖励。评价性完成家务，按照完成度以五级评分权重赋予奖励** | **在数据库存储中，为家务表设置‘奖励数目’字段，在编辑任务时填写。在UI界面，任务完成时，使用Rating组件为任务打分，根据打分情况（5分制），用分数/5 \* 奖励数目，并完成后为完成者‘鲜花数目’加上本家务的奖励数目** | | **支持为家务添加闹钟提醒，在任务指定完成时间到来时刻进行，在手机任何位置（无论本APP是否打开）弹窗和发声提醒，点击弹窗后跳转到APP首页** | **使用鸿蒙代理提醒业务，应用退至后台或进程终止后，系统会代理应用做相应的提醒。根据家务提醒类型，使用代理提醒中基于日历的提醒功能：在家务表项中，存储有家务预完成时间，将预完成时间设置为日历提醒的年月日时分，将家务名设置为提醒标题，将与完成时间time作为提醒id。设置后台代理后，在任务指定完成时间到来时刻进行，在手机任何位置（无论本APP是否打开）弹窗和发声提醒，点击弹窗后跳转到APP首页** | | **支持消耗鲜花抽奖** | **使用画布Canvas组件，绘制转盘，自定义奖励** |   **更多亮点的实现逻辑请将在后续【亮点功能实现】中做着重介绍** | | | | | | | | | |
| 1. 开发和运行环境（可能包括开发平台、运行平台、开发工具、第三方组件使用情况、其他支撑软件运行的条件等）   开发平台：DevEcoStudio  开发语言：ArkTS  运行平台：DevEcoStido中的鸿蒙模拟机 | | | | | | | | | |
| 三、设计说明（写明设计思想、程序的结构、功能设计、界面设计、模型设计、程序主要执行流程图，最后是核心源代码，截图等） 目录 3.1 代码结构  3.2 APP概览  3.3 数据库构建  3.3.1 数据库控制层面  3.3.2 数据库数据层面  3.4 闹钟提醒实现  3.5 家务奖励功能实现  3.6 抽奖功能实现  3.7 页面设计  ========================================================  **3.1 代码结构**  ├── entry/src/main/ets // 代码区  │ ├── agency  │ │ └── pages  │ │ └── Index.ets // 主页  │ │ └── CarvasPage.ets // 抽奖页面  │ │ └── JiatingIndex.ets // 家庭页面  │ │ └── PersonEdit.ets // 家庭成员管理页面  │ │ └── taskEditPage.ets // 家务编辑页面  │ │ └── TaskIndex.ets // 家务列表  │ ├── constants  │ │ └── ColorConstants.ets // 公共颜色常量  │ │ └── CommonConstants.ets // 公共常量  │ │ └── StyleConstants.ets // 公共样式常量  │ ├── database  │ │ ├── rdb // 数据库封装类  │ │ │ ├── RdbHelper.ets // 数据库帮助类接口  │ │ │ ├── RdbHelperImp.ets // 数据库帮助类实现  │ │ │ ├── RdbUtils.ets // 数据库工具类  │ │ │ └── TableHelper.ets // 表格帮助类  │ │ └── tables // 数据表  │ │ ├── DayInfoApi.ets // 日信息数据表  │ │ ├── FormInfoApi.ets // 表单信息数据表  │ │ ├── GlobalInfoApi.ets // 全局信息数据表  │ │ ├── TaskInfoApi.ets // 家务信息数据表  │ │ └── PrayInfoApi.ets // 奖励信息数据表  │ ├── utils  │ │ ├── Logger.ets // 日志记录工具  │ │ └── Utils.ets // 工具类  │ │ └── PersonList.ets // 家庭成员全局列表  │ ├── entryability  │ │ └── EntryAbility.ets // 程序入口类  │ ├── model // model  │ │ ├── DatabaseModel.ets // 数据库model  │ │ ├── RdbColumnModel.ets // 数据库列模型  │ │ └── WeekCalendarModel.ets // 日历model  │ └── viewmodel // viewmodel  │ ├── ColummnInfo.ets // 列信息模型  │ ├── DayInfo.ets // 日信息模型  │ ├── DrawModel.ets // 绘图模型  │ ├── FillArcData.ets // 填充弧数据模型  │ ├── FormInfo.ets // 表单信息模型  │ ├── PersonInfo.ets // 家庭成员信息模型  │ ├── PrayInfo.ets // 奖励信息模型  │ ├── TaskInfo.ets // 家务信息模型  │ ├── ReminderItem.ets // 提醒事项模型  │ └── WeekCanladerInfo.ets // 周历信息模型  └── entry/src/main/resources // 资源文件目录  **3.2 APP功能概览**  下面是站在用户角度，使用本APP考虑的调用顺序，列出了各种操作的流程图。  用户使用本APP主要完成三种操作：家庭成员管理，家务事项和抽奖。    图 1 APP功能概览  其中，添加家务流程图：    图 2 添加家务流程图  操作家务流程图：    图 3 操作家务流程图  查询家务流程图：    图 4 查询家务流程图  抽奖流程图：    图 5 抽奖流程图  **3.3 数据库构建**  **数据库构建代码相较于前几个实验，进行了代码结构上的优化，这也是本实验的核心亮点之一。**  前几次实验数据库插入，都是在UI界面直接调用鸿蒙数据库接口插入，没有做数据有效性检测，可拓展性也较差。  本实验中，将数据库操作分为了**控制层面和数据层面**，并各自编写不同的类，存放在不同的文件夹中以供维护。  3.3.1 数据库控制层面  在最底层，基于鸿蒙数据库操作建议接口，编写RdbHelper.ets，在内部定义RdbHelper接口，规定了数据库操作必须的函数，包括创建数据库，创建表，对表的增删改查，直接执行SQL语句等。然后，在RdbHelperImp.est对该接口进行类的实现。实现过程中，会对关键信息进行打印，例如RdbHelperImp类的saveData——像表中插入数据，就会调用’ @ohos.data.relationalStore’中rdbRestore的insert接口，并对data进行打印。    图 6 最底层数据库插入数据代码  然后，编写RdbUils类，相较于RdbHelperImp，这个类更完善的定义了数据库操作的流程，将一些必须合在一起的操作放在一个函数中，例如将关键操作例如创建数据库和插入表，绑定在一起，然后返回promise——作为操作成功与否的标志，因此用户通过这个类知道底层数据库操作的相关信息。  接着，就是对表操作的封装。对于每个表，都有和用户进行交互的tableApi，里面封装的是用户在UI界面中实际调用的函数。包括对用户传入数据处理成数据库可识别的bucket，并且将数据库返回的数据封装成为用户可识别的对象。总的来说，表项Api函数的功能逻辑共性为：  接收用户传入数据 -> 检验合法性 -> 提取传入数据的各有用项，将其转换为bucket -> 调用RdbUils，接收返回值 -> 对返回值进行处理 -> 通过callback接收用户对结果的操作  因此，数据库的调用层级抽象如图所示，图中也包含了每一层的任务和特点：    图 7 数据库控制层面的调用层级  3.3.2 数据库数据层面  为每一个数据库中存储的表编写类，封装其属性，表中的列必须在类中有相应的属性。在本APP中，保存的表包括：家务表，奖励事件表，家庭成员表，表概览如下：    图 8 家务表    图 9 奖励事件表    图 10 家庭成员表  编写TaskInfo, PersonInfo, PrayInfo封装这些表信息，同时作为ui界面经常传参的参数。以PersonInfo为例：    图 11 家庭成员类  数据单元封装后，需要联合在一起作为数据库操作数据层面的核心。编写RdbColumnModel.ets，定义每一个数据表表项title，并规定表项属性——包括是否非空，是否为主键，是否自增长。    图 12 RdbColumnModel部分代码  对表项属性也进行了封装，ColumnInfo封装了表项名，数据类型，长度，是否非空，是否为主键，是否自增长。    图 13 ColumnInfo类  对以上数据部分封装后，数据库控制层面的调用就只需要对数据层面封装的类进行调用即可。如果需要扩展，也只需要在现有框架下对应部分增加，修改代码，大大提高了程序的可维护性和可扩展性。在日后其他应用的开发中，也可以使用这一套数据库结构，避免重复造轮子的问题。  总之，本APP构建的数据库框架如下：    图 14 本APP构建的数据库框架  另外，为了方便用户UI对数据的访问不用每次都查询数据库，在第一次查询家务和家庭成员后，数据会被保存到内存中，其中：   * 家庭成员不经常被修改，且全局不同位置经常访问，编写PersonList.ets全局类，保存家庭成员信息，用Map<string,PersonInfo>维护，其中string意为家庭成员姓名 * TaskList经常变动，但会被各种组件访问，因此在UI界面中用@Provide修饰家务数组TaskList，在子组件中用@Consume访问   为了更好的组织家务状态，为家务状态添加标签，同样存入数据库，如图所示是家务的状态图：    图 15 家务状态图  **3.4 闹钟提醒实现**  闹钟提醒基于鸿蒙代理提醒业务，应用退至后台或进程终止后，系统会代理应用做相应的提醒。  根据家务提醒类型，使用代理提醒中基于日历的提醒功能：在家务表项中，存储有家务预完成时间，将预完成时间设置为日历提醒的年月日时分，将家务名设置为提醒标题，将与完成时间time作为提醒id。  设置后台代理后，在任务指定完成时间到来时刻进行，在手机任何位置（无论本APP是否打开）弹窗和发声提醒，点击弹窗后跳转到APP首页。  首先封装闹钟提醒类ReminderService，作为用户UI调用代理服务的接口，其中比较关键的函数是添加提醒函数：  public addReminder(alarmItem: ReminderItem, context: Context, callback?: (reminderId:number)=>void){  let notifyId: string = alarmItem.notificationId.toString();  this.hasPreferencesValue(context, notifyId, (preferences: preferences.Preferences, hasValue: boolean) => {  if (hasValue) {  preferences.get(notifyId, -1, (error: Error, value: preferences.ValueType) => {  if (typeof value !== 'number') {  return;  }  if (value >= 0) {  reminderAgent.cancelReminder(value).then(() => {  this.processReminderData(alarmItem, preferences, notifyId);  }).catch((err: Error) => {  Logger.error(Const.REMINDER\_AGENT\_TAG, `cancelReminder err: ${err}`);  });  } else {  Logger.error(Const.REMINDER\_AGENT\_TAG, 'preferences get value error ' + JSON.stringify(error));  }  });  } else {  this.processReminderData(alarmItem, preferences, notifyId);  }  });  }  用户在调用时，传入alarmItem——提醒的相关信息，这个信息被封装为AlarmItem类，其中包括了提醒标题，年月日时分，点击提醒后跳转的APP和Page等信息：  export default class ReminderItem {  hour: number = 0;  minute: number = 0;  year: number = 2023;  month: number = 1;  day: number = 1;  title: string = '';  content: string = '';  notificationId: number = 0;  }  然后，该函数根据回调函数，在提醒添加成功时返回提醒ID。  函数处理逻辑，将通知ID转换为字符串，用作Preferences的键值。通过hasPreferencesValue方法检查Preferences中是否已存在该通知ID对应的值。如果已存在值，获取该值，如果为正数，说明已存在相应提醒，取消之前的提醒，然后处理新的提醒数据。如果Preferences中不存在该通知ID对应的值，直接处理新的提醒数据。  其中，hasPreferencesValue用到了首选项，保存提醒事项的ID。  在UI层面，例如——家务编辑完成后，前端需要综合提醒相关属性，得到AlarItem，调用addReminder接口，并通过try-catch处理返回值和异常：  try {  let publishReminderInfo = new ReminderItem();  publishReminderInfo.hour = Number(params?.time.split(':')[0]);  publishReminderInfo.minute = Number(params?.time.split(':')[1]);  publishReminderInfo.year = Number(params?.date.split('-')[0]);  publishReminderInfo.month = Number(params?.date.split('-')[1]);  publishReminderInfo.day = Number(params?.date.split('-')[2]);  publishReminderInfo.title = params.taskName  publishReminderInfo.content = params.beizhu  publishReminderInfo.notificationId = new Date().getTime()  console.log(`提醒年 ${publishReminderInfo.year} 月 ${publishReminderInfo.month} 时分 ${publishReminderInfo.hour} : ${publishReminderInfo.minute} id ${publishReminderInfo.notificationId}`)  this.reminder.addReminder(publishReminderInfo, context,(newId)=>{params.taskID = newId;Prompt.showToast({message:'添加提醒成功'})});  } catch (error) {  Logger.error('publishReminder', JSON.stringify(error));  }  最后，得到的效果如图：    图 16 任务编辑时开启提醒    图 17 提醒通知  **3.5 家务奖励功能实现**  在数据库存储中，为家务表设置‘奖励数目’字段，在编辑任务时填写。在UI界面，任务完成时，使用Rating组件为任务打分，根据打分情况（5分制），用分数/5 \* 奖励数目，并完成后为完成者‘鲜花数目’加上本家务的奖励数目    图 18 Rating组件  根据打分情况修改可获得鲜花数目：用分数/5 \* 奖励数目，并向下取整。在rating发生变化时候，使用@Watch修饰，每次变化调用onRatingChanged函数    图 19 使用@Watch修饰Rating    图 20 根据打分情况修改可获得鲜花数目  将结果存入家务对应表项，即可永久保存数据。  **3.6 抽奖功能实现**  抽奖功能借鉴了CodeLab中转盘抽奖的示例代码，但也进行了更改和创新。首先将抽奖事件和用户鲜花数进行绑定，在抽奖页面中加载出家庭成员信息，每一次抽奖消耗5朵鲜花。  加载成员信息，获取屏幕宽高，在aboutToAppear中：  window.getLastWindow(context)  .then((windowClass: window.Window) => {  let windowProperties = windowClass.getWindowProperties();  this.screenWidth = px2vp(windowProperties.windowRect.width);  this.screenHeight = px2vp(windowProperties.windowRect.height);  })  .catch((error: Error) => {  Logger.error('Failed to obtain the window size. Cause: ' + JSON.stringify(error));  })  for(let val of PersonList.personMap.values()){  this.personNameRange.push(val.personName)  }  在页面即将显示时，获取当前窗口的尺寸并转换为视觉像素单位，同时初始化抽奖人名单。获取窗口尺寸的操作通常用于适应不同屏幕尺寸，而初始化抽奖人名单则为后续用户选择抽奖人提供了候选名单。  设置开始抽奖事件：  if(this.person.personValue<5){  Prompt.showToast({message:'余额不足,再去做点家务吧'})  } else{  this.enableFlag = !this.enableFlag;  this.startAnimator();  this.person.personValue -= 5  PersonInfoApi.updateDataByName(this.person,()=>{})  PersonList.updatePerson(this.person)  }  在用户进行抽奖时，首先检查用户的余额是否足够支付抽奖费用（5个鲜花），如果足够，则切换抽奖按钮状态，启动抽奖动画，并扣除相应的鲜花费用。同时，更新用户信息到数据库和全局用户列表。如果余额不足，显示提示信息提示用户去做家务来获取鲜花。  对于转盘内容，编写绘制抽奖转盘的Canvas绘图类DrawModel，该类通过包括外圆花瓣、外圆小圆、内圆、内部扇形区域、扇形区域文字、奖品图片等元素的绘制，打造界面。其中包括draw 方法，绘制填充的圆弧的 fillArc 方法，以及各种绘制外观元素的方法，如绘制花瓣、外圆小圆、内圆、内部扇形区域、文字和奖品图片等。通过使用常量、枚举和工具类，以及对资源的获取，实现了可配置、可定制的转盘绘制效果。  在drawArcText和getPrizeData中，静态设置奖励名和奖励说明：    图 21 绘制奖励名（圆盘扇形中文字）    图 22 奖励说明  但是，只静态配置奖励不能完全满足用户需要，我们还是希望能动态的设置奖励，包括消耗的鲜花数目等等。但是由于时间原因在第一版本的APP中没有做，但是我仍然预留了很多接口，例如在数据库中建立了【奖励】表：    图 23 数据库中的奖励表详情  在下一步开发本APP中，将允许用户动态配置这些奖励，可加载到抽奖圆盘，也可直接用鲜花进行兑换。  **3.7 界面设计**  界面设计追求组件化的原则，很多组件设计了阴影：    图 24 组件的阴影效果  很多组件也加入了属性动画，在Width变化时会触发动画：  图 25 家务关键词搜索框的属性设置（包括动画）  所有页面设计如下：    图 26 APP页面设计 | | | | | | | | | |
| 五、总结及分析（完成过程中的心得体会，成功与失败之处，经验，收获，建议、将来计划和展望）  最后，是我的心得体会。以下都是我手打的肺腑之言：  感谢老师和助教带我入门了移动应用软件开发，在鸿蒙系统旭日东升之时，让我赶上了ArkTS开发的时代潮流。  要说移动应用软件开发带给我最大的体会是什么，我想就是两个字——应用。  我是计算机科学与技术专业的学生，在我的专业课中，接触到的都是非常底层的东西，包括计算机组成与原理，操作系统，编译原理，就算是机器学习也是比较上层的了。而移动应用开发让我接触到的是作为计算机设计者最上层的东西，是直接与用户交互的东西，在开发的过程中，我用到了数据库，用到了后台代理，用到了云开发，用到了各种迭代方法，系统API调用——种种我曾经在专业课中学到的技术。当我真正去作为开发者使用这些技术的时候才明白这些技术每一个真正的使用场景，和优势。  例如，在使用数据库的时候，不可能任何一个信息都存入数据库——那会带了非常大的访问开销和空间开销，我会考虑如何存储表信息才能最大程度的节约空间，要不要用到数据库关系设计中的范式来优化数据存储。再比如，在程序开发的全局变量中，我会考虑去维护什么数据结构，能让我在我想要用到的地方方便的调用。  当今吗，伴随着鸿蒙开发课堂在B站上对大众开放，不少社会人士也加入了鸿蒙开发的行列。我在B站上分享了我开发软件的日常，就碰到了很多陌生网友找我要代码。我看到这些网友很多都是大专的。我想我们985大学生和他们的优势在哪里——就在于我们现在开发过程中考虑到了各种优化方案，我们通过数据库专业课学到的知识优化数据库管理，我们用软件工程学到的知识确定软件功能是切合所需，并且软件体系结构设计关注扩展性和维护性，而不是写成**屎山代码**，我们用数据结构预算法的知识更好的组织应用数据。**只有充分发挥我们专业课的长处，才能在鸿蒙开发的潮流中脱颖而出。**  不过，移动应用开发遇到的难处我也是经历了很多，很多很多，几乎70%的时间都在调试bug。经过一学期的学习，我也有了自己调试bug的经验。  像arkTs这种声明式语言，和传统C或者Java调试很不一样，前端页面出现错误极有可能是向前端的问题也有可能是后端的问题，首先检查后端，将需要渲染的数据通过console.log打印出来（最好用JSON.stringfy直接输出整个类），没有问题，再去找前端的问题。  此外，软件开发还有一个重要的经验就是，很多东西必须提前就考虑清楚——这个功能我是作为一个component子组件，还是另外做一个page；这个数据，我是用@state直接修饰为这个组件独有的，还是用@Link和父组件双向同步，还是用@Provide - @Consume实现跨组件绑定。如果不提前考虑清除，后面遇到功能扩展的时候就会话很大的功夫来改。  最后要说一点就是，软件开发不仅是编程，还是艺术——所有的软件最后都是和用户交互，因此页面设计无比重要。我在开发软件的过程中，总会想到很多**曾经使用过的APP**，例如美团，微信等等，看看他们的设计理念——什么样的组件搭配，什么样的颜色搭配。  大致成功和失败的收获就是上述吧。然后未来的打算——我是打算一直搞下去的，我很喜欢自己用开发工具开发出一个自己能用的小玩意。也希望自己开发的软件能够至少被家人使用，我的目标就是设计各种家庭APP，给自己的爸妈用，老爸老妈老了，他们用到的APP很多都不会用。我是最东他们的人，我可以设计出最贴合他们需求的APP，这就是我的目标。  再次感谢老师和助教的培养，希望我能在开发软件的道路上多多努力，持之以恒！ | | | | | | | | | |