

**《软件工程》项目报告**

**题目： 资源调度平台**

**课程名称： 软件工程**

**专业班级： CS2008**

**同组成员： 学号：U202015546**

**姓名： 符诗章**

**学号：U202015519**

**姓名： 马腾飞**

**学号：U202015530**

**姓名： 刘鸣**

**学号：U202015529**

**姓名： 刘明炜**

**指导教师： 瞿彬彬**

**报告日期： 2022.11.23**

**计算机科学与技术**

**任 务 书**

**一 总体要求**

1. 综合运用软件工程的思想，协同完成一个软件项目的开发，掌软件工程相关的技术和方法；

2. 组成小组进行选题，通过调研完成项目的需求分析，并详细说明小组成员的分工、项目的时间管理等方面。

3. 根据需求分析进行总体设计、详细设计、编码与测试等。

**二 基本内容**

根据给出的题目任选一题，自行组队，设计与开发中软件过程必须包括：

1. **问题概述、需求分析：**正确使用相关工具和方法说明所开发软件的问题定义和需求分析，比如NABCD模型，Microsoft Visio，StarUML等工具 (20%)；
2. **原型系统设计、概要设计、详细设计**：主要说明所开发软件的架构、数据结构及主要算法设计，比如墨刀等工具（35%）；
3. **编码与测试**：编码规范，运用码云等平台进行版本管理，设计测试计划和测试用例（30%）；
4. **功能创新**：与众不同、特别吸引用户的创新（10%）；
5. **用户反馈**：包括用户的使用记录，照片，视频等（5%）。

**目 录**

[1 问题定义 3](#_Toc120573246)

[1.1 项目背景与意义 3](#_Toc120573247)

[1.2 项目基本目标 4](#_Toc120573248)

[1.3 可行性分析 4](#_Toc120573249)

[1.4 人员管理和项目进度管理 4](#_Toc120573250)

[2 需求分析 5](#_Toc120573251)

[2.1 UML相关需求分析图 5](#_Toc120573252)

[2.2 原型系统设计 6](#_Toc120573253)

[3 概要设计和详细设计 11](#_Toc120573254)

[3.1 系统结构 11](#_Toc120573255)

[3.2 类图 13](#_Toc120573256)

[3.3 关键算法设计 14](#_Toc120573257)

[3.4 数据管理说明 16](#_Toc120573258)

[4 实现与测试 18](#_Toc120573259)

[4.1 实现环境与代码管理 18](#_Toc120573260)

[4.2 关键函数说明 18](#_Toc120573261)

[4.3 测试计划 19](#_Toc120573262)

[4.4 结果分析 28](#_Toc120573263)

[5 总结 29](#_Toc120573264)

[5.1 用户反馈 29](#_Toc120573265)

[5.2 全文总结 30](#_Toc120573266)

[6 体会 31](#_Toc120573267)

[7 附录 34](#_Toc120573268)

# 问题定义

## 项目背景与意义

### 用户需求

销售或者都买闲置物品的需求，满足学生群体二手书及闲置物品交换的需要或者在疫情情况下解决物资交易不及时不方便的问题。

### 解决用户需求的做法

1. 每一个用户都可以进行注册登录。
2. 用户可以通过选择社群在自己的区域中进行资源调度交换。
3. 用户可以通过分类和搜索更易地发现自己想要的物品。
4. 用户可以联系卖家进行线下交易。
5. 用户同时也可以在平台上发布自己的闲置物品。

### 产品给用户的好处

解决了闲置物品信息不对等和更新不及时的问题。在小的社群之间，以往闲置资源交换往往采用qq群或者微信群中约定的方式进行交换，这导致昨天发布的闲置物品信息，在几天后便难以找到，且需要的交换的人不知道物品是否仍在，使得信息的有效性和有效期限降低。通过本应用提高了物品交换的效率。

### 市场竞争

当前市面上存在咸鱼等二手交易平台，但是咸鱼等平台面向全国人民，交易过程较为繁琐。在小的社群如社区中，由于距离较近，因此采用线下交换等方式更佳，不需要太过繁琐的操作。因此，一个小的资源置换平台更能满足这类用户的需求。

### 产品推广

通过在宿舍楼栋间推广，逐步代替宿舍间的各种二手交易群。

## 项目基本目标

我们经过前期讨论后决定，采用具有多种优势的微信小程序的方式完成此次课题设计，使用github来管理项目代码，利用Android Studio工具进行开发。用户能够添加商品到自己的商品库，并根据商品库中的商品发布帖子。用户可以创建社群，并且在社群中发布帖子，帖子是属于该社群的。用户也可以根据关键字模糊搜索社群，对于搜索命中结果社群，能够加入到社群中。对于已经加入的社群，用户能够进入社群查看社群中的帖子，也可以查看帖子详情。帖子详情中有发帖人的相关联系信息。

## 可行性分析

根据前期我们所讨论出的NABCD模型来评估，我们的项目在解决这类需求的产品里有一定市场，并且通过比较来看，我们的项目相比之下也有更好的特色，比如有针对性，简洁，免费等等，所以在市场这一块我们针对的用户需求是比较理想的。

此外，我们选择的方式是Android软件，开发工具比较成熟，学习成本相应较低。这学期开设数据库课程，方便应用联网。而且对于界面等设计，我们小组有很大的能力优势，所以可实施性较高

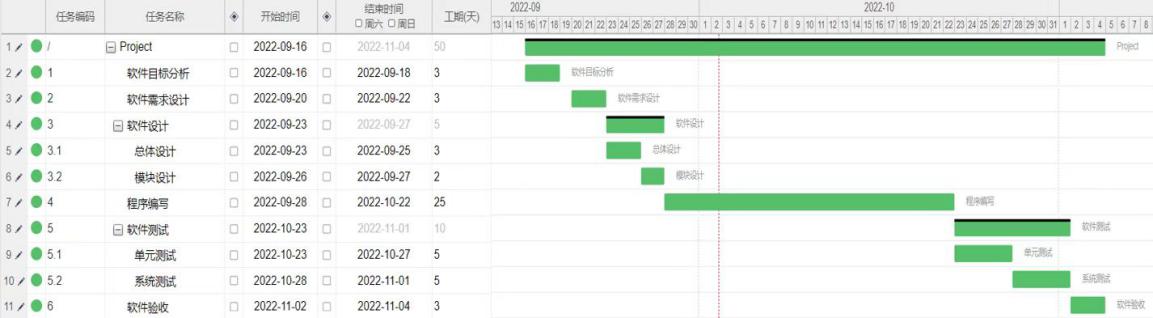
## 人员管理和项目进度管理

符诗章：前端UI设计，网络连接和加速，页面后端开发

马腾飞：数据库JDBC，图片导入与压缩

刘鸣：页面后端开发，数据库JDBC，前端UI设计

刘明炜：页面后端开发，前端UI设计



# 需求分析

## UML相关需求分析图

|  |
| --- |
| UML用例图 |
| 图2.1 UML用例图 |

## 原型系统设计

### 设计工具

使用在线设计工具：墨刀免费版

### 设计成果

启动界面如下图，显示软件版本，软件名称以及软件简介

|  |
| --- |
| 1_启动页 |
| 图2.2 启动界面图 |

软件启动后进入登录界面，在登录界面输入账号和密码进行登录

|  |
| --- |
| 1-1_登录 |
| 图2.3 登录界面图 |

如果是初次使用，需要先注册账号

|  |
| --- |
| 1-1-1_注册页 |
| 图2.4 注册界面图 |

登录成功之后进入首页，在首页进行推送，显示部分用户可能需要的帖子

|  |
| --- |
| 1-2_首页_待售 |
| 图2.4 首页显示图 |

在社群界面显示我加入的社群，

|  |
| --- |
| 1-3_社群 |
| 图2.6 社群显示图 |

通过模糊匹配进行搜索发现社群，然后加入到搜索到的社群

|  |
| --- |
| 1-3-5_社群搜索 |
| 图2.7 搜索社群 |

除了通过搜索加入社群以外，也可以创建自己的社群

|  |
| --- |
| 1-3-4_创建社群 |
| 图2.8 创建社群 |

在社群内部可以发布有关商品的帖子，帖子包括商品图片、商品价格、商品简介，以及发布者的联系方式

|  |
| --- |
| 1-2-2_好物详情 |
| 图2.9 帖子信息 |

每个用户有一个商品库，可以向商品库中添加商品

|  |
| --- |
| 1-4_发布_书 |
| 图2.10 添加商品 |

能够显示关于用户的信息

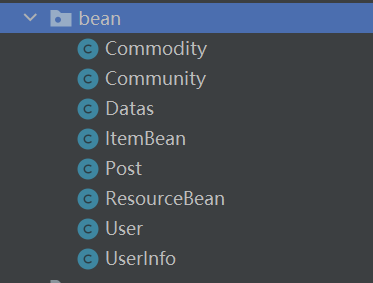
|  |
| --- |
| 1-5_我的 |
| 图2.11 我的信息 |

# 概要设计和详细设计

## 系统结构

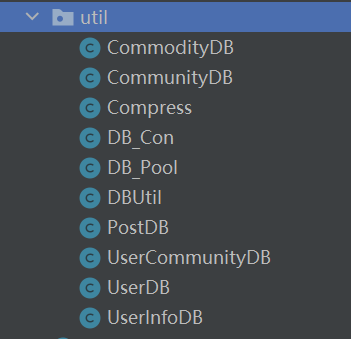
1. bean包：

主要用于声明各种实体类，例如User用户类、Community社群类、Post帖子类等，在每个实体类的内部，定义了其私有变量、构造函数和get与set方法。



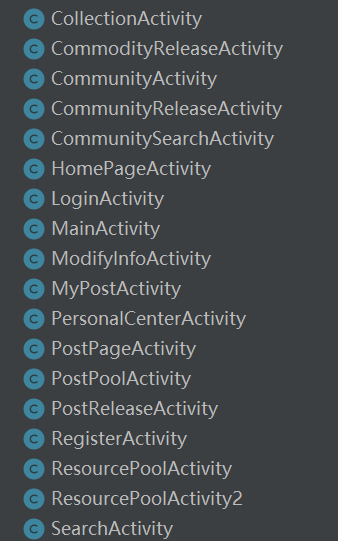
1. util包

主要用于声明各种数据库工具类，如CommodityDB商品数据库、UserDB用户数据库等，数据库工具类的内部是该表中涉及到的操作方法。数据库的具体实现详见本报告中的数据库部分。



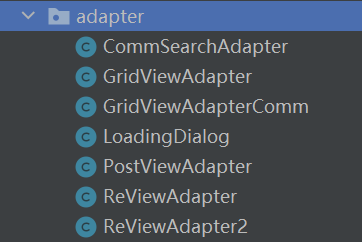
1. activity包

activity是一个应用程序组件，提供一个屏幕，用户可以用来交互为了完成某项任务.其中的所有操作都与用户密切相关，是一个负责与用户交互的组件。



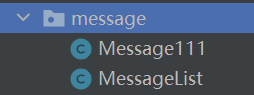
1. adapter包

主要用于实现适配器类。Adapter是用于连接后端数据和前端显示的适配器接口，是数据data和UI（View）之间一个重要的纽带。本软件大量使用到RecyclerView等，需要用到不同的Adapter。



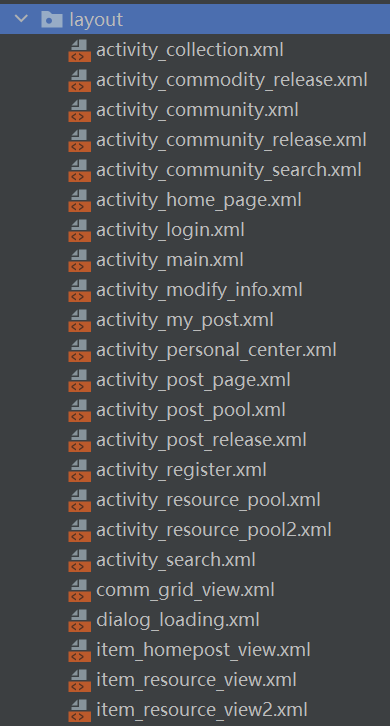
1. message包

主要使用单例模式，用于存储事件之间传递的共享数据。这种实现方法相较于在事件中利用bundle传递更高效且健壮性强，相较于Android自带的广播机制BroadcastReceiver，这种方法更简洁轻量



1. xml布局文件

使用XML布局文件来定义每个activity事件的用户界面，在文件中通过XML属性对所有组件进行控制。本软件主要使用约束布局ConstraintLayout，同时也针对不同界面功能灵活使用了相对布局和线性布局。



## 类图

本软件主要有四大实体类，分别是用户类、商品类、社群类、帖子类，每个类都有其主要的私有变量和相关方法，具体细节和各个类之间的关系详见如下类图。

|  |
| --- |
| IMG_256 |
| 图3.1 系统类图 |

## 关键算法设计

### 图片压缩算法

由于我们并没有对用户上传的图片的大小加以限制，所以当用户上传较大的图片时，可能会产生较大的时延。同时，用户在获取包含较大图片的数据时，也会产生较大的时延。为此，我们考虑进行质量压缩。质量压缩的原理是通过改变图片的位深和透明度来减小图片占用的磁盘空间大小，可以用于想保持图片质量的同时减小图片所占用的磁盘空间大小，通过bitmap.compress()方法即可较为方便的实现。另外，为了进一步限制图片的大小，我们还进行了尺寸压缩，最终将图片的大小控制在100KB以内。经过测试，图片压缩可以有效地加快加载速度，且图片在用户手机上的清晰度较高，说明图片的质量也得到了保证。

### 线程通信

1. 使用线程的理由：

在主线程中进行网络操作会带来巨大的阻塞，可能会给软件带来UI的卡顿，更有可能使得软件崩溃。同时在android 4.0 以上，网络连接的相关操作不能放在主线程中。因此，我们将网络请求的相关操作放入线程中执行。

1. 线程通信的实现：

在软件的具体设计中，线程通信通过使用Thread类与Handler类得以实现。通过Thread类创建新的线程以发起网络请求并更新数据，在网络数据到达后，通过使用Handler类来向主线程传递对应的messeage，以通知主线程根据到达数据做出响应，如根据到达的数据重新渲染UI。通过引入线程通信，我们可以有效地减少卡顿并大大地提高软件的用户体验。

1. 线程通信的优势：

在引入线程通信前，我们仅仅通过Thread来实现网络请求，即在Thread类中发起网络请求，但完全不向主线程传递数据到来的信息。在这一模式下，为了保证用于渲染UI的数据正确，所有涉及数据的相关操作都必须在线程执行完毕后后再执行。因此我们使用如图所示的思路，使用thread.join()方法强制栓塞线程，等待线程执行完毕后再进行下一步操作。这无异于是抛弃了线程的优势，将多线程强制转换为了同步执行的单线程，使得软件的执行效率大大降低，同时也带来许多糟糕的用户体验，如跳转明显卡顿和页面跳转可能导致黑屏等等。

在引入线程通信后，我们使用Handler类在线程之间传递信息，从而提高了UI渲染等操作的执行效率。在发起网络请求但数据尚未到来的过程中，我们首先根据数据为空的情况渲染出当前的大致界面。在数据到来后，我们借助Handler类向主线程传递信息，主线程接到对应信息后，再根据到来的数据重新渲染出正确的UI。具体的软件执行流程如下图所示。通过引入Handler机制，我们很好地优化了用户体验，减少了界面跳转的卡顿，并且规避了跳转黑屏的问题。

### 数据库连接池

1. 原先的数据库连接模式:

通过观察软件的运行日志我们发现，在一次数据获取的过程中，程序实际上并非只向数据库发起了一次连接，而是在频繁地创建与释放连接。有时可能只是执行一次查询操作，程序就重复地创建和销毁了好几次与数据库的连接connection，这个过程为程序带来了很大的开销，同时也使得数据的传输效率降低。因此为了避免连接的频繁创建与销毁，我们引入数据库连接池的机制来提高软件的运行效率。

1. 使用数据库连接池的原因:

数据库连接池是在程序启动时建立多个数据库连接，并将这些连接组成一个连接池，统一进行管理，由程序动态地对池中的链接进行申请，使用，释放。数据库连接池具有以下优点: 1.资源复用，避免连接的重复创建与销毁。2.带来更快的系统响应速度与数据请求速度。3.实现统一的连接管理，避免数据库连接泄露。4.避免多个线程同时使用一个连接。基于数据库连接池的以上优点，我们引入数据库连接池。

1. 数据库连接池的具体实现:

由于安卓开发的特殊性( javax包中的部分函数无法使用 )，我们无法直接使用网络上开源的优秀数据库连接池，所以在软件开发中，我们通过实现JDBC中提供的连接池接口 javax.sql.Dataource 来得到数据库连接池类DB\_Pool。

连接列表: 在DB\_Pool中维护一个列表pool以存储提前申请的连接。由于连接池可能在多个线程中被使用，列表必须能够保证线程安全，所以我们使用Collections.synchronizedList方法创建列表。

连接初始化: 实现静态方法init用于初始化连接，为了在软件启动时初始化连接，本方法应该在login界面初始化时就被使用。应注意地是，初始化连接的过程应当在线程中执行，否则会导致连接创建失败。

获取连接: 实现getConnection方法用于获取连接，当列表不为空，从列表中取出连接交给使用者，否则，为其重新申请一个连接。

释放连接: 实现release方法用于释放连接，当方法被调用时，我们将连接归还到线程池中，若此时线程池中连接数过多，我们则将连接释放。

连接维护: 启动定时器定期重新创建新的连接以代替旧的连接，防止长期不使用的连接被数据库切断而失效。

## 数据管理说明

在TOS宇宙的软件设计中，数据库是软件的核心部分，所有涉及数据管理方面的功能对应了数据库中的相关操作。具体来说，用户对个人信息、商品、社群、帖子的操作都由对数据库中相应表的增、删、改、查实现。TOS宇宙的数据库主要由6张表构成：商品表Commodity、社群表Community、帖子表Post、用户表User、用户信息表UserInfo、用户社群表UserCommunity。每张表的属性如下图所示。

|  |
| --- |
|  |
| 图3.2 数据库中表的属性 |

我们对于数据库的设计，保证了数据库中的每个关系模式都满足鲍依斯-科得范式（BCNF），有效地减少了冗余信息，成功地避免了修改异常、删除异常等问题。

在将数据库设计落实到软件过程中，我们经历了两个阶段：离线阶段和在线阶段。在设计软件的初期，我们出于易于实现的考虑，首先实现了软件的离线版本，即将数据保存在本地，以更方便地、更快速验证数据库设计的可行性。在此阶段，我们使用安卓自带的SQLite，利用其自给自足、无服务器、零配置、事务性、轻量级等特点，成功地将数据保存在了用户的手机上。在确认功能正确且基本完备后，我们的软件进入了第二个阶段——在线阶段。为了将数据保存在云端，我们购买了腾讯云中的MySQL数据库，并通过JDBC实现了软件的云数据库链接。但是，由于商品和社群类存在图片，简单地使用JDBC实现联网会导致内容加载过慢，获取一张几兆大小的图片往往需要5s以上。所以我们针对软件的数据库连接进行了一系列优化，成功将加载时延基本控制在1s以内。这一部分的将在算法设计部分详细说明。

# 实现与测试

## 实现环境与代码管理

1. 实现环境

操作系统：

PC端：Windows10，Windows11，Linux

移动端：Android， Harmony

开发平台：AndroidStudio

1. 代码管理

采用Github平台管理项目：



Github地址为<https://github.com/Fuzhang1/317software.git>。

## 关键函数说明

1. private void initData()

每个界面需要根据不同的条件发起网络请求获得需要显示的数据，网络请求在线程中进行，数据到来时向主线程传递信号

1. protected void onCreate(Bundle savedInstanceState)

界面初始化函数，跳转到该界面时自动执行

1. public void onClick(View view)

对于Android布局中的控件，根据其需求，定义相应的点击事件

1. private void showGrid(boolean isVertical, boolean isReverse)

当主线程收到数据到来的信号时，根据请求数据，使用相关适配器，渲染相关布局如商品卡片等

1. public void release(Connection con)

释放连接，将连接返还到连接池中维护，当连接池中连接过多时，直接销毁连接

1. public Connection getConnection()

获取连接，从连接池中取出一条连接使用，而非重新申请

## 测试计划

### 常用测试方法

软件测试方法的目的包括：发现软件程序中的错误、对软件是否符合设计要求，以及是否符合合同中所要达到的技术要求，进行有关验证以及评估软件的质量。最终实现将高质量的软件系统交给用户的目的。而软件的基本测试方法主要有静态测试和动态测试、功能测试、性能测试、黑盒测试和白盒测试等。

软件测试方法众多，比较常用到的测试方法有等价类划分、场景法，偶尔会使用到的测试方法有边界值和判定表，还有包括不经常使用到的正交排列法和测试大纲法。其中等价类划分、边界值分析、判定表等属于黑盒测试方法；只对功能是否可以满足规定要求进行检查，主要用于软件的确认测试阶段。白盒测试也叫做结构测试或逻辑驱动测试，是基于覆盖的全部代码和路径、条件的测试，通过测试检测产品内部性能，检验程序中的路径是否可以按照要求完成工作，但是并不对功能进行测试，主要用于软件的验证。

基于本次设计的是资源调度平台的Android-App，采用真机测试的方法，即黑盒测试，对App进行虚拟机测试和真机测试。

### 手机测试

在虚拟机中测试页面跳转，显示相关页面的信息，以及数据库功能等，具体的测试过程如下：

1. 注册登陆界面显示

|  |  |
| --- | --- |
|  |  |
| 图4.1 登录界面测试 | 图4.2 注册界面测试 |

如果已经拥有账号，直接用账号登录；如果已经拥有账号，需要先注册账号，然后用注册的账号登录。

1. 登录界面提示：（登陆成功与失败）

|  |  |
| --- | --- |
|  |  |
| 图4.3 登录成功界面测试 | 图4.4 登录失败界面测试 |

如果账号存在且与密码相匹配，则登录成功；如果账号不存在或者密码不正确，则只提示登录失败

1. 新手引导教程：

对于新注册的用户，首页会有一个名为“新手教程”的帖子，点开帖子查看帖子详情会有新手教程

|  |  |
| --- | --- |
| 66b1bf67a9200db240f18de8a034b3f | cd5946d42525fd72ff3992316d79ea0 |
| 图4.5 首页新手教程帖子 | 图4.6 新手教程帖子详情 |

1. 个人信息显示

|  |
| --- |
|  |
| 图4.7 个人信息显示测试 |

在本界面显示用户的个人信息，包括头像，昵称，QQ以及电话，图片可以从手机内相册选择，点击保存后保存更改后的信息。

1. 我的界面显示

|  |
| --- |
|  |
| 图4.8 我的界面显示测试 |

我的界面即个人中心，包括个人信息，我的帖子和退出登录。点击退出登录后，软件退出，下次使用软件需要重新登录。

1. 我的帖子显示

|  |
| --- |
|  |
| 图4.9 我的帖子显示测试 |

在本界面，显示所有自己发布过的帖子。

1. 首页推送帖子功能测试

|  |  |
| --- | --- |
|  |  |
| 图4.10 首页推送功能测试1 | 图4.11 首页推送功能测试2 |

在首页界面显示10个帖子，如果帖子数目不足10个的情况，显示所有帖子。根据推送算法，显示最多10个帖子。点击帖子可以查看帖子详情。

1. 添加商品功能测试

|  |
| --- |
|  |
| 图4.12 添加商品功能测试 |

在自己的商品库里添加商品，商品的相关信息有名称、描述和图片。

1. 帖子详情界面显示

|  |
| --- |
|  |
| 图4.13 帖子详情界面测试 |

在帖子详情界面显示帖子的详细信息，包括商品的价格、名称、图片，和发帖人的联系方式。

1. 商品列表显示

|  |
| --- |
|  |
| 图4.14 商品库显示测试 |

在商品库中显示自己添加的所有商品，删除后的商品不会在本界面显示。

1. 删除商品功能测试

|  |  |
| --- | --- |
|  |  |
| 图4.15 全选删除商品测试 | 图4.16 部分选择删除商品测试 |

可以通过全选删除所有商品，也可以部分选择来删除商品，点击“垃圾桶”图标即可删除

1. 我加入的社群显示

|  |
| --- |
|  |
| 图4.17 我加入的社群界面显示测试 |

在本界面显示所有我已经加入的社群，对于每个社群，点击进社群可以查看社群内的所有帖子

1. 创建社群功能测试

|  |
| --- |
|  |
| 图4.18 创建社群测试 |

1. 搜索社群功能测试

|  |  |
| --- | --- |
|  |  |
| 图4.19 默认搜索界面测试 | 图4.20 模糊搜索测试 |

在搜索界面，搜索前默认显示所有社群，在输入关键字进行模糊匹配后，显示搜索的命中结果社群

1. 加入社群功能测试

|  |
| --- |
|  |
| 图4.21 加入社群功能测试 |

在显示搜索结果社群后，点击“加社群”加入社群，字样变为“已加入”，在我加入的社群可以显示加入的社群。

1. 社群内帖子显示

|  |  |
| --- | --- |
|  |  |
| 图4.22 社群内帖子显示测试1 | 图4.23 社群内帖子显示测试2 |

在社群内部显示发布于该社群的所有帖子，在社群内部可以发布帖子，也可以退出社群。

1. 发布帖子功能测试

|  |
| --- |
|  |
| 图4.24 发布帖子测试 |

在每个社群内部可以发布属于该社群的帖子，商品从自己的商品库中进行添加。发布后的帖子可以在该社群内显示，对其他社群不可见。

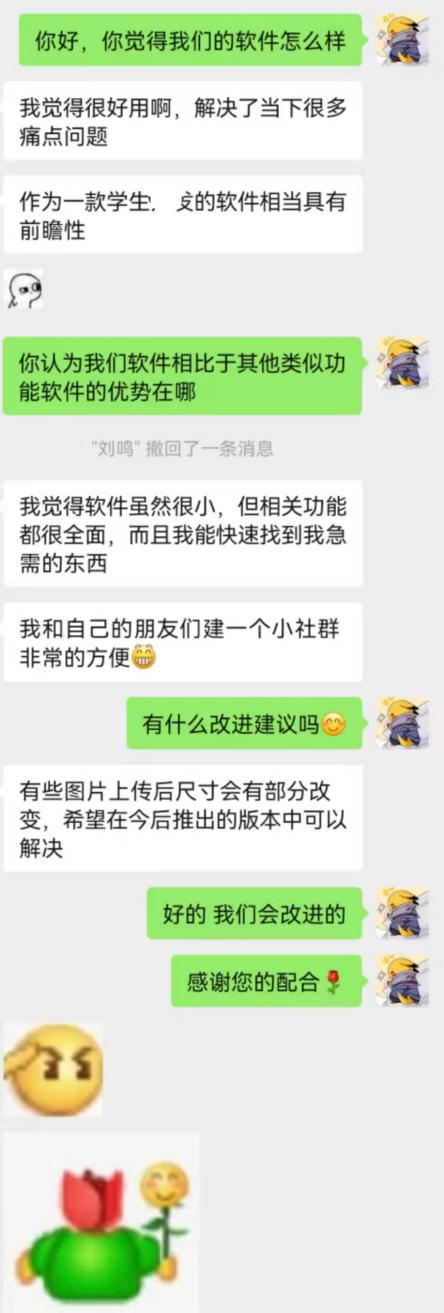
## 结果分析

软件实现了网络连接，能够访问服务端数据库。通过测试，按钮能进行正常的页面跳转操作，并且跳转的逻辑正确。除此之外，各种页面和提示信息都能正常显示和工作。

通过以上测试结果，能够添加商品到自己的商品库，并根据商品库中的商品发布帖子。用户可以创建社群，并且在社群中发布帖子，帖子是属于该社群的。用户也可以根据关键字模糊搜索社群，对于搜索命中结果社群，能够加入到社群中。对于已经加入的社群，用户能够进入社群查看社群中的帖子，也可以查看帖子详情。帖子详情中有发帖人的相关联系信息。

# 总结

## 用户反馈



用户反馈分析：

通过用户反馈可知，我们软件功能比较全面，能够实现用户所需要的功能，包括发布信息以及快速搜索所需要的物品。不足之处是图片上传后的尺寸有所改变，与用户期望的图片有一点差别。在后续我们需要对这个问题做出改进，使用户上传的图片与显示的图片保持一致。

## 全文总结

在软工项目开始前，选题的时候，挑选比较贴近生活的项目。考虑到疫情期间，居家隔离的时候，许多人面临物资严重不足的问题，缺粮缺菜，而有的人这些物资会有剩余。所以我们选择开发的项目叫资源调度平台，以解决物资供需不平衡的问题。

我们选定Android软件作为项目的形式，利用Android Studio进行开发。开发思路是先开发一个无法联网的本地版本；然后在本地版本的基础上进行联网实现。也正如我们的预期，经过数周的时间，编写并调试完成了本地版本。然后用一周的时间完成联网实现，但发现联网会导致软件运行十分缓慢，之后用了一周的时间进行了优化和加速。

虽然与市面上流行的软件相比还有很多不足之处，但我们预期的功能全部实现。首页能够推送给用户帖子用户能够添加商品到自己的商品库，能够创建并加入社群。用户能够根据关键字搜索并加入社区，能够在自己加入的社群中，根据商品发布帖子。我们的特色之一是，同一个商品在不同社群里可以有不同的价格。对于已加入的社群，用户能够查看发布在该社群中的所有帖子，挑选出自己想要的商品，在帖子中查看卖家的联系方式完成线下交易。

# 体会

1. 符诗章同学，分工:前端UI设计，网络连接和加速，页面后端开发，体会：

在本次软件工程的课程中，我学会了Android应用程序的相关开发。刚开始进行软件开发时，由于Android开发中前端UI的开发与web前端大不相同，导致我对于UI设计一筹莫展。但后来通过观看网上的教学视频并结合web前端的相关知识进行学习，我渐渐地掌握了安卓布局的相关知识。在一开始，我们只能做出只适配电脑模拟器的UI界面，到了手机上就会显得比例失调。后来，我通过学习百分比布局实现了适配于多数手机屏幕尺寸的UI界面。除了前端UI设计，本次课程中，我还学习了安卓中事件类和适配器类的写法，为app中的各个界面加上了点击、跳转和加载等处理逻辑，使得页面更加灵活并能投入使用。在项目中，由于我对于安卓开发的不熟练，写出了许多有bug的代码块，但通过观察日志，我们最终解决了bug，使得程序能够正常的使用。在app使用jdbc成功连接上数据库之后，我们发现了软件在加载大文件时传输会非常慢的问题，这无疑会使得用户体验大大降低。因此我又学习了线程和数据库连接池的相关知识，从线程优化和连接的提前缓存的两个角度优化了大文件的加载速度，最终使得程序能够正常使用。

总而言之，在本次软件开发的过程中，我学习了安卓前端、事件类与适配器类、jdbc、网络连接等新知识，同时又通过实践加深了对于线程和java知识的巩固。同时，在本次课程中，我也体验了软件开发的完整过程，并且明白了提前进行完备设计的必要性及与团队成功及时沟通解决问题的重要性，使我受益匪浅。

1. 马腾飞同学，分工: 数据库JDBC，图片导入与压缩，体会:

在本次《软件工程》课程的学习中，我了解了软件工程的概念，系统地学习了UML和面向对象的软件开发方法，以及软件开发的需求、设计、实现、测试等。此外，我在本次课程中还体验到了完整的软件开发过程，和同组成员一起开发了一款功能较为完善的资源调度APP——TOS宇宙。在开发TOS宇宙的过程中，我主要负责数据库，图片导入与处理的部分。在软件开发的初期，我参与了数据库中表格的设计，并使用SQLite实现了对这些表格的增删改查，并使用数据库实现了账号查重、密码验证等功能。在软件开发的后期，我成功实现了手机图库图片的导入，并且使用JDBC代替SQLite，成功连接了腾讯提供的云数据库，实现了软件的网络连接。在经过一段时间的软件测试与体验后，我们发现过大的图片会导致软件的严重卡顿，甚至长时间没有响应。于是，我引入了图片压缩算法，成功地控制了图片的大小，较为显著地改善了图片加载速度，提升了用户体验。

总的来说，我在本次《软件工程》课程的学习中收获了很多，既系统地学习了软件开发的相关知识，也体验了软件开发的完整过程，同时也体会到了团队协作在软件开发中的重要性。

1. 刘鸣同学，分工：页面后端开发，数据库JDBC，前端UI设计，体会:

本次软件工程课程让我收获良多。我在本次项目中的多个部分都有承担一定的开发任务，比如后端实现、前端设计、数据库实现等。一个项目从无到有的建立，需要多个方面共同努力。让我感触很深的是代码规范的重要性，在代码长度急剧增加的情况下，不规范的代码很有可能会给后续的工作带来困难，在代码优化的过程中，必须确保代码能够正常工作，然后再考虑代码的优化，优化时也尽量维持代码可读性。其次，在本次项目中我还接触到了许多新知识，准确来讲大部分的开发对我而言都是全新的、需要学习的。我逐渐掌握了如何快速从各个渠道找到现有问题解决方案的办法，对于新的技术问题也可以利用已掌握的知识进行新的思考。很多之前在理论上理解的知识，例如数据库相关的操作，在本次项目中也通过实战得到了检验，也让我更加深刻的意识到理论知识和实践是有一定距离的，还需要继续努力去掌握。

在一个需要多人协同的大型项目上，需要大家群策群力对项目的整体规划做出事先的安排，而不要等到项目设计到一半再来改需求、改设计框架，这样就会事倍功半，造成人员精力的浪费。在今后的程序设计过程中，要多多注意运用软件工程这门课上所学到的知识，在程序设计的初始阶段，就对整个程序有着清晰的目标与规划。总的来说，本次项目让我受益匪浅。

1. 刘明炜：页面后端开发，前端UI设计,分工:

这次软件工程项目我主要负责的是页面后端开发，前端UI设计。由于没有Android开发经验，一开始是无从下手。好在之前学过Java课程，入门并不是特别的困难。前期花了许多的时间学习Android开发的相关知识，包括Activity，适配器等。在写完部分代码进行调试的时候，在虚拟机上总会出现软件崩溃的情况，调试起来也很麻烦。总是一点一点确定bug的位置，然后发现都是一些不起眼的问题，比如控件的id拼错等。在进行调试的时候，多亏了队友的火眼金睛，发现很多低级错误。在前端UI设计中，一开始设计的很简略，想法是先正常实现功能，后期再进行优化。前期的工作很简单，但这也为后期巨大的工作量埋下了伏笔。在基本功能实现之后，后期花了很多时间对界面进行优化。除此之外，也发现了代码规范在编写项目中的重要性。对于关键变量和复杂函数要有注释说明；函数、变量以及类的命名要规范，让自己看得懂，也要让别人看得懂。也体会到了使用git进行版本管理的优越性。

这次软件工程项目，让我深切体会到了项目制作的复杂程度。小组成员都需要随时沟通，每个人不仅要负责自己的部分，还需要和其他人的工作进行协调。最后我们每个人还是圆满地完成了任务，或许有一些小瑕疵，但整体结果还是不错的，我相信这次的经历也会对我以后的工作提供一定的经验。

# 附录

由于项目基于AndroidStuidio开发，具有很多java类与xml文件，全部放在附录中展示并不现实，因此此处只展示软件的部分关键函数代码。整个项目的代码可以在电子版与git仓库中查看。

软件的部分关键函数代码如下:

//用于模拟数据

private void initData() {

//创建数据集合

loadingDialog.show();

mData = new ArrayList<>();

int community\_id = this.getIntent().getIntExtra("community\_id",0);

mData2 = new ArrayList<>();

Thread th1= new Thread(new Runnable() {

@Override

public void run() {

mData = PostDB.postsByCommunity(community\_id);

for(Post x : mData)

{

mData2.add(CommodityDB.findCommodity(x.getCommodity\_id()));

}

Message message =Message.obtain();

message.what=1;

mhandler.sendMessage(message);

}

});

th1.start();

}

private void showGrid(boolean isVertical, boolean isReverse) {

GridLayoutManager layoutManager = new GridLayoutManager(this, 1);

layoutManager.setOrientation(isVertical ? RecyclerView.VERTICAL : RecyclerView.HORIZONTAL);

layoutManager.setReverseLayout(isReverse);

mList.setLayoutManager(layoutManager);

adapter = new PostViewAdapter(mData,mData2);

mList.setAdapter(adapter);

initListener();

}

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_post\_pool);

loadingDialog = new LoadingDialog(this);

bundle = this.getIntent().getExtras();

user\_id = this.getIntent().getIntExtra("user\_id",0);

mList = (RecyclerView) this.findViewById(R.id.recycler\_re\_view3);

btn\_submit\_re = findViewById(R.id.btn\_submit\_post);

btn\_fanhui=findViewById(R.id.btn\_fanhui3);

tv\_title = findViewById(R.id.tv\_comm\_title);

tv\_quit = findViewById(R.id.tv\_quitCom);

String com\_name = this.getIntent().getStringExtra("com\_name");

initData();

showGrid(true,false);

tv\_title.setText(this.getIntent().getStringExtra("com\_name"));

btn\_fanhui.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent();

intent.setClass(PostPoolActivity.this,CommunityActivity.class);

intent.putExtras(bundle);

startActivity(intent);

finish();

}

});

btn\_submit\_re.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent = new Intent();

intent.setClass(PostPoolActivity.this,PostReleaseActivity.class);

intent.putExtras(bundle);

startActivity(intent);

}

});

tv\_quit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

loadingDialog.show();

Thread th= new Thread(new Runnable() {

@Override

public void run() {

int comm\_id = CommunityDB.searchByTitle(com\_name);

PostDB.deletePostByQuit(user\_id, comm\_id);

UserCommunityDB.deleteUserCommunity(user\_id, comm\_id);

}

});

th.start();

try {

th.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

loadingDialog.dismiss();

Intent intent = new Intent();

intent.setClass(PostPoolActivity.this,CommunityActivity.class);

intent.putExtras(bundle);

startActivity(intent);

}

});

}

public void release(Connection con){

Log.wtf("2","release:" + pool.size());

if(pool.size()>max){

try {

con.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

else {

pool.add(con);

}

}

@Override

public Connection getConnection(){

Log.wtf("1","get:" + pool.size());

if(pool.size()>0){

return pool.remove(0);

}

else{

return DB\_Con.getConn();

}

}