第2组-Lab3实验报告

1 小组分工

1.1 人员安排

- 组织：林琰钧
- 数据库：林琰钧、马成
  - 使用Navicat进行Mysql数据库关系模型设计
  - 部署数据库
- 后端：林琰钧、马成
  - SpringBoot
  - 编写controller、service、mapper，以及必要的vo、dto、entity
- 前后端接口：林琰钧、马成、王兆瀚、李咸若
  - 使用APIFox进行前后端接口设计
  - 包括url、请求参数（xxxFVO）、响应参数（xxxVO）
- 前端：王兆瀚、李咸若
  - Vue
  - 根据前后端接口，设计页面
- 测试：马成、李咸若
  - 编写数据库数据脚本
  - 编写单元测试
  - 进行接口测试
- 部署：李咸若
  - 将项目前后端串联并部署

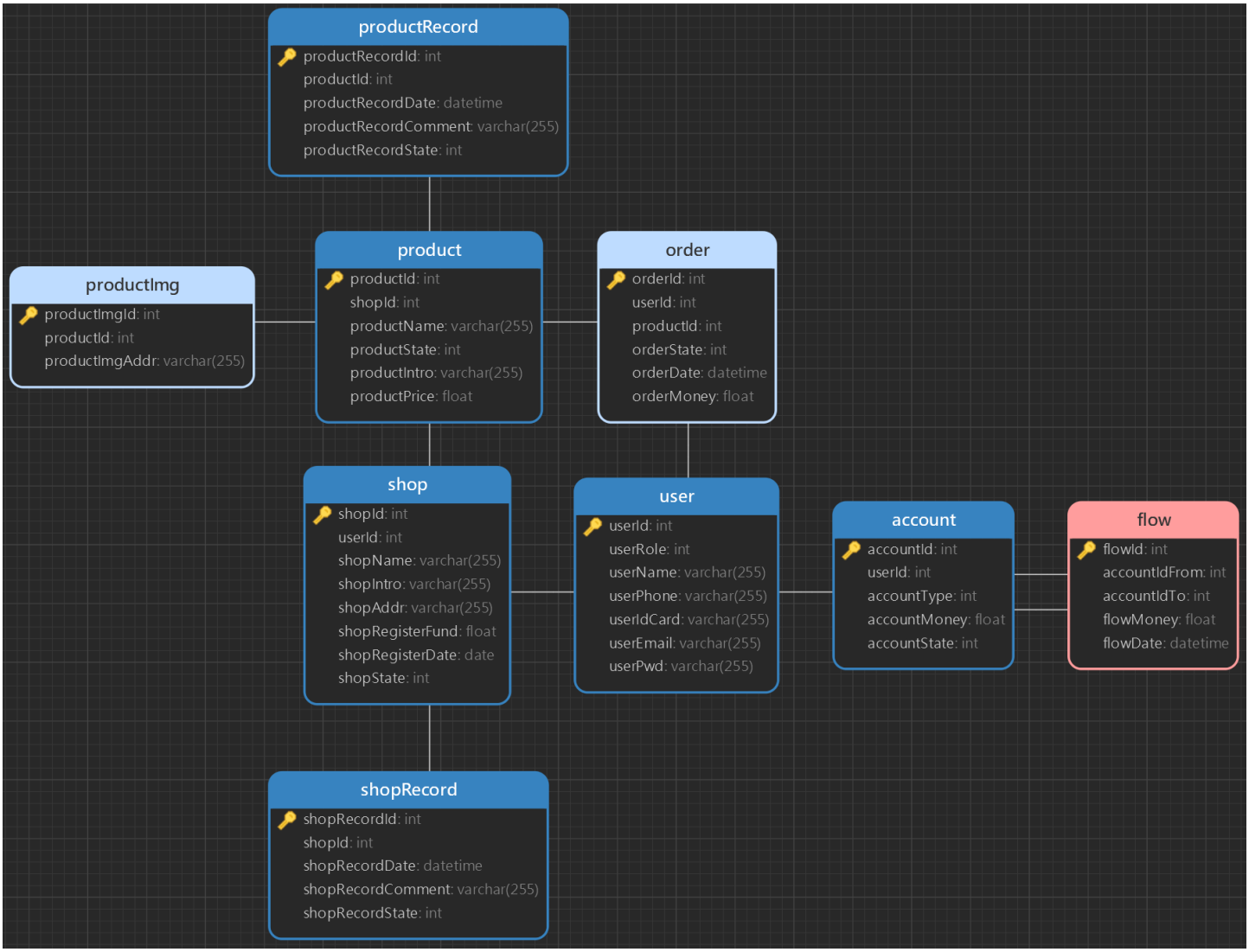
1.2 开发记录

- 4.13 (Thu.)
  - 下午算法课下课讨论数据库设计
- 4.14 (Fri.)
  - 下午AI课下课讨论数据库设计

- To 4.14 (Fri.) 23:59.p.m.
  - 林琰钧、马成、王兆瀚：利用Navicate设计lab3数据库关系模型、导出sql文件
  - 完成数据库设计
- To 4.16 (Sun.) 23:59.p.m.
  - 王兆瀚：设计前端静态页面
- 4.17 (Mon.)
  - 下午编译课下课讨论API设计
- 4.18 (Tue.)
  - 晚上逸夫楼501讨论API设计
- 4.20 (Thu.)
  - 算法课下课讨论API设计
- To 4.20 (Tue.) 23:59.p.m.
  - 林琰钧、马成、王兆瀚、李咸若：利用APIFox设计前后端接口与VO
  - 完成API设计
- To 5.5 (Fri.) 23:59.p.m.
  - 林琰钧、马成：后端开发
    - 林琰钧：user、account、flow
    - 马成：product、shop、order
  - 王兆瀚、李咸若：前端开发
    - 王兆瀚：user、account、product
    - 李咸若：admin

2 实验设计

2.1 数据库与类图



- 上述设计中
  - 蓝色表示不能删除entry，但可以增加、修改
    - user本身没有注销接口，不允许删除
    - 所有的record作为归档记录，不希望被删除
    - product、shop、account有外键，删除会导致级联删除，而我们不希望看到record、flow和order中的字段被删除；因此，若非要删除的话，其删除表示为State字段的一个特定值
  - 白色表示可以删除entry
    - Img只能增加不能修改
    - order可以修改State
  - 红色表示可以增加，不能删除、修改
  - 有关图片的信息都存储在后端的某个文件夹，数据库存储图片地址
- user
  - 存储用户、商户、管理员的信息
  - userRole

        - 用户0
        - 商户1
        - 管理员2
    - 特殊id
        - 钱包：userId为1，userName为“wallet”，只是用来占坑的，作为外键引用的对象，login会特判不允许它登录
        - 管理员：userId为2，userName为“root”
- shop
    - 存储商店信息
    - shopState
        - 审核待通过0
        - 审核未通过1
        - 商店开放2（包括删除未通过）
        - 删除待通过3
        - 已删除4
- product
    - 存储所售卖的商品类别
    - productState
        - 审核待通过0
        - 审核未通过1
        - 已上架2（审核通过）
        - 已下架3
        - 已删除4
- account
    - 账户表
    - accountType
        - 个人账户0
        - 商店账户1
        - 商城利润2
        - 商城中间账户3
        - 虚拟账户4，唯一一个，用于充值的流水表accountIdFrom的外键引用

    - accountState
        - 有效0
        - 删除1
    - 特殊的id
        - dummy账户：accountId是1，userId是1。有无限多的钱，充值提现都与该账户有关
        - 中间账户：accountId是2，userId是2
        - 商城利润账户：accountId是3，userId是2
- flow
    - 流水表，用于转账
- order
    - 订单表
    - 把用户订单、商家订单、购物车都以订单表形式表示，表示状态的orderState不同
    - orderState
        - 未付款（购物车里）0
        - 已付款未收货1
        - 已付款已收货2
- shopRecord是商店申请记录
    - shopRecordState
        - 开店申请未处理0，商店申请一经提交不允许修改
        - 开店申请通过1
        - 开店申请拒绝2
        - 删除申请未处理3
        - 删除申请通过4
        - 删除申请拒绝5
- productRecord是商品申请记录
    - productRecordState
        - 未处理0，此时如果修改商品信息直接覆盖时间戳
        - 上架申请通过1
        - 上架申请拒绝2
- 数据库脚本
    - cd到/resources/static/design/lab3/lab3-test.sql 所在的文件夹

- mysql -u root -p <lab3-test.sql

2.2 商城逻辑

- 用户（user）
  - 用户注册
    - 检查注册信息规范（同lab2）
    - 注册成功则生成个人账户
  - 用户信息修改
    - 不能修改角色、身份证号，其他的修改也需要检查规范
    - 修改成功则生成新token给前端
  - 用户密码修改
    - 需要输入原密码和新密码，独立于用户信息的修改
    - 修改成功则生成新token给前端
  - 用户登录
    - 生成token给前端（同lab2）
  - 用户退出
    - 啥也不做（同lab2）
- 账户（account）和流水（flow）
  - 获取账户信息
    - 给后端传一个账户类型
    - 后端通过token获取用户，检查是否有该类型的账户并返回
  - 转账
    - 充值和提现是特殊的转账
      - 充值从dummy账户转账；提现转账到dummy账户
      - dummy账户的金额始终不变，但个人账户或商店账户金额会改变
    - 商店注册申请与审批、商店删除申请与审批、商品购买付款与发货都会涉及与中间账户之间的转账
    - 每次转账都会插入一条flow流水表
  - 流水表
    - 给后端传一个账户类型
    - 后端通过token获取用户，检查是否有该类型的账户并返回出账或入账该账户的流水表

- 逛街（visit）
  - 逛商品（product）
    - 默认分页展示所有在售商品
    - 点击某个商品就会进入该商品的页面
      - 可以查看商品的详细信息
      - 可以加入购物车或直接购买（详情见order）
  - 逛商店（shop）
    - 在商品页面中可以跳转查看商品所属的商店的详细信息
    - 在商店详细信息中可以跳转分页查看该商店的所有在售商品
- 订单（order）
  - 用户（user）
    - 商品页面中
      - 可以直接购买，即生成一个已付款未收货的订单；买家个人账户资金转移至中间账户
      - 可以加入购物车，如果购物车没有该商品时，则生成一个未付款的订单
    - 购物车页面显示所有未付款订单
      - 传输购物车列表的时候会传输商品状态，如果商品处于下架状态则不允许用户购买
      - 可以购买，即将未付款的订单变成已付款未收货的订单；买家个人账户资金转移至中间账户
      - 可以删除，直接删除订单
    - 物流页面显示用户所有已付款未收货订单
    - 交易完成页面显示用户所有已收货订单
  - 商家（owner）
    - 物流页面显示商家所有已付款未收货订单
      - 点击发货，即将已付款未收货的订单变为已付款已收货的订单；中间账户资金转移至商店账户
    - 交易完成页面显示商家所有已付款已收货订单
- 商户（owner）
  - 商店（shop）
    - 商店注册
      - 商店注册时，提交注册，注册资金从虚拟账户转移到中间账户，在管理员审核之前，商店信息都不允许修改，并且不能添加商品

        - 如果管理员拒绝商店注册，则注册资金从中间账户转移到虚拟账户，此时商户可以修改商店信息，之后重新提交审核，视为重新走了一遍注册流程
        - 如果管理员同意注册，则生成一个商店账户，注册资金从中间账户转移到商城利润账户，之后商店信息就不再允许修改了，此时可以添加、修改、删除商品
    - 商店的删除
        - 检查商品是否都已发货，如果有未发货则不允许删除；
        - 如果都已发货，则将商店状态变为删除待通过，所有已上架的商品状态变为已下架，已提交审核变成审核拒绝，原因是商家申请删除商店
        - 如果审核通过，则将所有未删除的商品转化为已删除，将商店账户的余额转移给虚拟账户，将商店的账户设置为已删除
        - 如果审核未通过，只要把已下架的的商品转化为已上架
  - 商品（product）
    - 商品列表包括审核待通过、审核未通过、已上架的商品
    - 添加商品只能添加商品的文字信息
        - 初始状态是审核待通过
        - 添加商品审核记录
    - 删除商品前需要保证商品都已发货
        - 把商品状态变为已删除
        - 删除与商品关联的所有图片
    - 修改商品前需要保证商品都已发货
        - 可以增加或删除图片，或修改文字信息，这两种修改是独立的
        - 修改会导致状态从审核未通过或已上架变为审核待通过
        - 添加商品审核记录
- 管理员（admin）
  - 商店（shop）
    - 审核申请注册的商店
        - 不通过，则按照商店注册失败流程
        - 通过，则按照商店注册成功流程
    - 审核申请删除的商店
        - 不通过，则按照商店删除失败流程
        - 通过，则按照商店删除成功流程
  - 商品（product）

        - 审核申请上架的商品
            - 不通过，则商品状态变为审核未通过
            - 通过，则商品状态变为已上架
            - 更改商品审核记录
- 申请记录（record）
    - 商店申请记录
        - 商店注册或删除操作都会生成状态为未处理的申请记录
        - 一经提交，在审核之前，都不允许对商店中的信息进行进一步的操作
    - 商品申请记录
        - 商品新增会生成状态为未处理的申请记录
        - 商品修改，如果没有该商品的申请记录，则会生成状态为未处理的申请记录；如果已经是审核待通过，则更新审核记录的时间戳
    - 审核后的记录状态会变为已处理
        - 如果管理员审核通过则会记录“审核通过”
        - 如果管理员审核未通过则会有未通过原因字段
        - 商店申请删除时，那些待审核的商品会自动未通过审核，会记录“申请删除商店”
    - 请求商店纪录
        - 如果是Admin获取所有商店记录
        - 如果是Owner获取没有删除的店铺所有商店纪录
    - 请求商品纪录
        - 如果是Admin获取所有商品记录
        - 如果是Owner获取没有删除的店铺所有商品纪录

2.3 测试用例

1. 对Mapper完成了单元测试，使用JUNIT框架以及Springboot自带的支持可以较快的完成测试
2. 对部分Service函数做了单元测试检验其可用性，但是对于一些需要用到Token的函数没有对Service层做另外的测试。
3. 对后端单独进行了测试，使用APIFOX的相关功能实现。
4. 编写了数据库测试语句，大致根据逻辑使用C++语言生成了一份逻辑自洽的数据库语句，插入了约500个用户350个商店和8000个商品。并根据这些商店商品的状态完善了申请纪录逻辑等。同时为了前端展示的方便讲商品的名字和简介设置成为了比较有逻辑的语句而不是随机的英文字符（除了这两个基本还都是随机的字符）。这里使用了ChatGPT来辅助生成商品名等。

3 问题及解决方案

3.1 主要问题

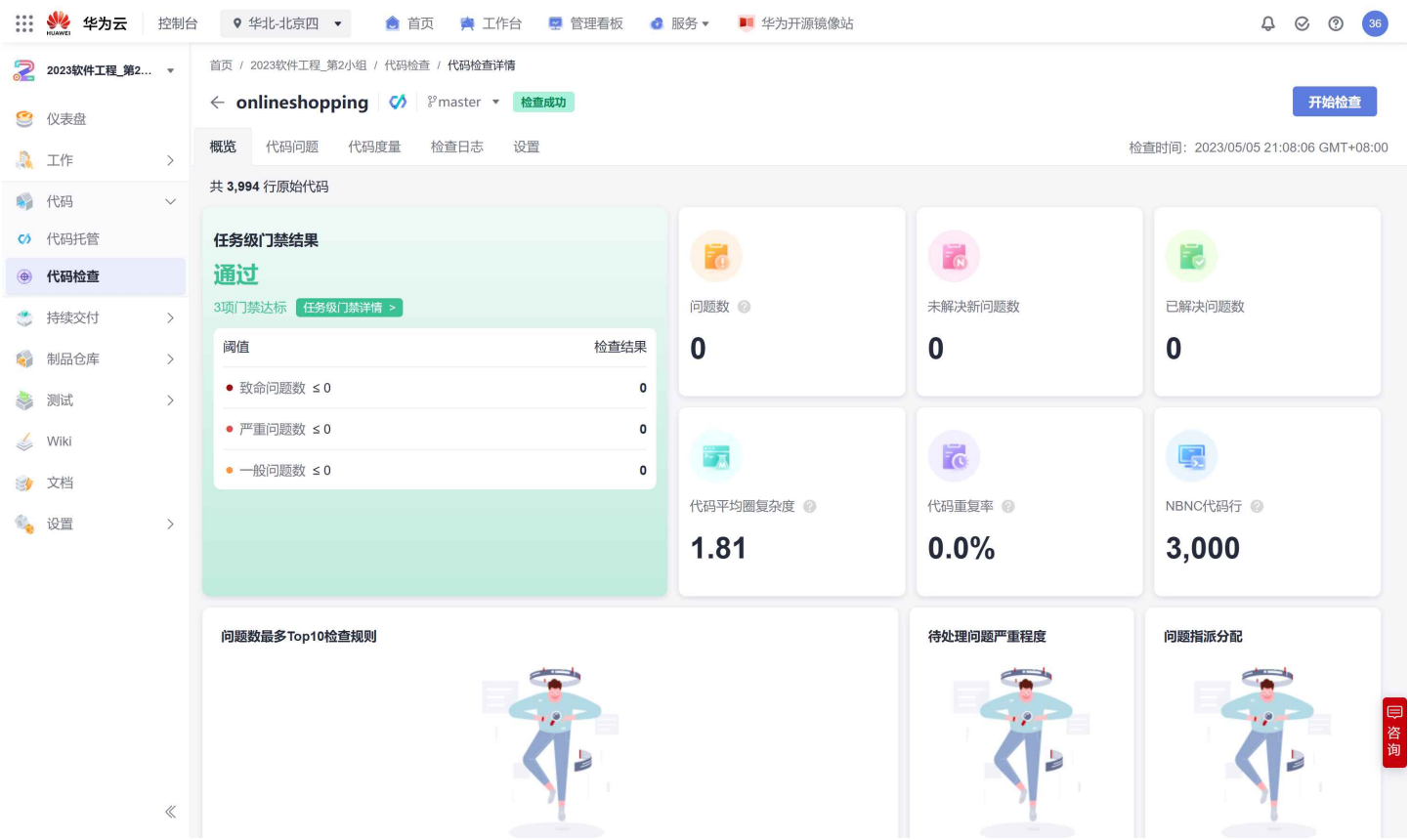
最主要的问题是数据库和API的设计，解决方案如上

3.2 次要问题

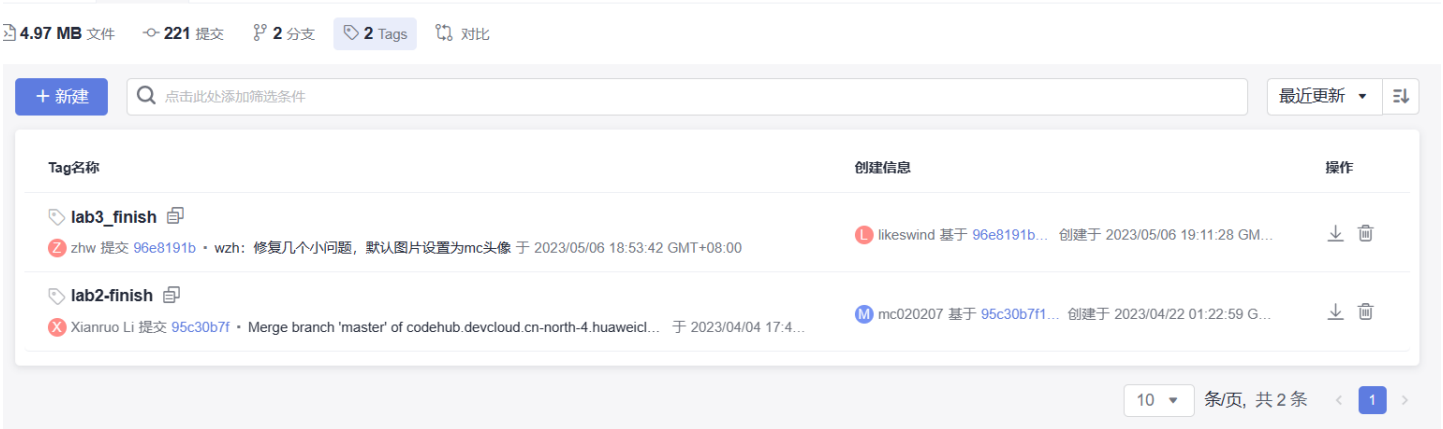
其他问题出在前后端交互上，列举如下

- 密码
  - 前端把密码md5加密再发给后端，后端就不进行md5加密了，这样不传输明文密码防止抓包
- 时间
  - 类型：string
  - 格式：YY-MM-DD HH:MM:SS
  - 注意时区是GMT+8
  - 相关链接：
    - mysql时区设置：https://blog.csdn.net/weixin_42825651/article/details/116519584
    - 前后端传输时间设置：https://blog.csdn.net/qq_36963762/article/details/117085922
- 图片
  - 前端上传后端
    - 类型：multipart
    - 前端图片以form-data表单形式传递，后端用MultipartFile接收并保存在"...(项目路径)/src/main/resources/static/img/xxx.jpg"，数据库存储url："img/xxx.jpg"
      - 数据传输格式设计与APIfox测试：https://www.cnblogs.com/yclh/p/16458758.html
      - 实现细节参考：https://blog.csdn.net/weixin_52065369/article/details/120412307
    - 后端需要做一个资源重定向，将"img/xxx.jpg"重定向到"...(项目路径)/src/main/resources/static/img/xxx.jpg"：https://blog.csdn.net/taiguolaotu/article/details/105405174
  - 后端返回前端
    - 类型：string
    - 后端给前端图片url "img/xxx.jpg"让前端显示，会通过上述资源重定向获取图片

4 代码检查结果



5 Tag截图



6 心得体会

6.1 林琰钧

我负责的内容是：

- 团队组织：进行人员分工与任务安排、制定开发计划、定期召开线下会议讨论进度
- 文档记录：编写项目开发的约定与安排文档、lab开发进度文档
- 架构设计：设计数据库、前后端接口
- 代码编写：编写部分后端代码

我的心得体会是：

- 第一次lab主要是学习技术栈，有了技术栈基础后，不论之后多少轮迭代，后续的开发基本就是砸时间造功能了，但有时候也会学习一些新技术，比如下载保存图片之类的
- 感觉我们的迭代式开发要变的东西好多，从底下的数据库到上面的接口都得变，同时前后端代码也加了一堆，相比于上次而言多了五六倍，开发时间太紧了，希望老师能多给点时间，比如一个lab给3~4周的开发时间（毕竟还有很多其他专业课也有很多pj要做qwq）

6.2 马成

我负责的内容是：

- 数据库模型设计
- 代码编写：编写所有有关商店、商品、订单及他们申请纪录的增删改的代码
- 测试：编写数据库测试文件以及完成单元测试

我的心得体会是：

- 感觉这次实验的功能的复杂性突然提升，需要花费更多的时间去完成数据库接口和逻辑的设计。但是目前看起来这些时间是比较有效并且是值得的。总体来说代码编写上还算比较顺畅，感觉没有很多的中途修改逻辑的时候我觉得这对代码编写来说是很好的。
- 感觉可以对已经讨论出来的逻辑大致做一个大纲，在写Service的时候可以大致对着大纲完成编写这样就不容易遗漏逻辑。由于逻辑的复杂要求对代码编写的精准度进一步提高。我认为这也是我将来编写代码需要注意的地方，严格控制笔误的可能性。同时需要考虑到较多的corner cases让代码整体更加可靠。
- 由于逻辑的逐渐复杂需要对每一个部分的测试做更加完善的准备。之前在做单元测试和做光后端测试的时候可能比较草率，第一没有测到很多的corner cases第二是没有仔细的观察数据的变化是否符合预期。导致有一些错误在最后联调之前都没有发现。其实都是一些小的笔误，我认为这是应该改进的，首先应该在编代码的时候就避免这些错误，在自己做测试的时候也应该更加严谨一些。

6.3 王兆瀚

我负责的内容是：

- 前后端接口设计
- 前端页面设计
- 代码编写：前端页面编写

我的心的体会是：

- 有了lab2的基础之后，虽然lab3的开发很快上手，但要学一些更深入的东西来完成复杂的视图逻辑。
- 页面的设计和审美有很大关系，而且要尽可能考虑用户的交互体验。

- 这次lab任务量很多，整个项目相当于重构了一次，迭代成本很大。如果有一个总的大纲，让项目有一个总的构想，我们就可以预留出一些接口，减小项目的迭代难度。
- 希望我们的服务器可以批下来，这样就可以测一些并发问题。

6.4 李咸若

我负责的内容是：

- 前后端接口设计
- 前后端串通及测试
- 代码编写：前端界面动态组件编写

我的心得体会是：

- 修改原先的项目比添加新功能难。应该预留扩展空间，而这种扩展空间不应局限在接口、界面的设计上。
- 命名的可扩展空间的重要性极高。不适应新功能的命名需要全部更改，会面临多版本不同命名逻辑的困扰，或者会需要机械性的将所有旧命名逻辑的东西更改，工作量大且枯燥。
- 以图片为代表的多态的信息传输比想象中难很多，需要深入去了解现有的方法。
- 随着项目的增大，测试的整个流程变得繁琐，且容易遗漏。对于整个功能的设置，可能更需要一个统一化的流程而非直觉使用下的测试。