Day1构思

想好了大概功能。

制作3级页面的不同原型图。

用gpt大概列出代码整体逻辑，需要的功能等。

Day2开始写代码，搭建分层架构框架

了解了分层架构，为自己的代码整了简单的分层架构。开始往里填东西。

第一次了解并配置了简单的虚拟环境（老费劲了）

写出了第一个完整的豆瓣爬虫，并一条龙的处理了数据。

大概了解了想要爬取gr不容易，需要更多步骤。

Day3重新设计功能实现方式，ip被封。

一直在微调代码框架。

了解和稍微使用了Git。

尝试n多反爬虫策略，ip被封一天。

发现很多功能无法实现，**改逻辑**，重画原型图。

书名爬取代码写了一半。

Day4阉割功能，再次重新设计逻辑

更理解了分层架构

**再次重构**功能并且阉割，search页面反爬严重，放弃。

第一次利用git退回原版本并创建分支。

Day5完善反爬，查询审核条件

学了2h的http协议视频，把session好好填完了。

发现爬虫应该过不了小程序审核，很慌，查了半天发现个人使用不上线就行。

改进数据提取逻辑，基本规避违规。

Day6再次转换搜索逻辑，功能阉割

第一次用api，尝试googlebook api，发现豆瓣无法提取英文名，所以无法获取isbn。

书名搜索无法做到。**阉割**，用用户输入豆瓣详情页url。

详情页——豆瓣英文页——英文isbn——gr搜索的逻辑。

创建了gr的爬虫，确定能爬到。

Da7爬取到了ajax存储的json评论

完成了gr的爬虫逻辑框架，基本爬虫驾轻就熟了。

了解了ajax，和爬ajax的方法，并爬到了未清洗的json。

数据提取不到，翻来倒去换代码，排查原因排查1h，发现是因为把max和min两个参数填反了，导致筛选出0数据……

Da8 成功爬取了所有需要的数据

成功爬取了豆瓣书籍信息，评论。Gr书籍信息，评论。

框架全部完成，内容基本填完。

散装数据结构待整理，一些返回空值的情况，文字换行情况待处理。

Day9摸了。纠错，简化代码

把提取评论的功能制作成了可重复利用的函数

增加了一些错误返回

Day10 改bug，调整代码结构，写接口

了解了flask封装api

因为缺少相关知识，做后端的时候没有考虑到接口，返回去**小改后端逻辑**，把数据进行了拆分，代码微调

写了一半接口。

发现一些清洗数据时的bug，修改。

Day11摸了

把intr的提取bug改了一下。

D12 摸了

虽然还是不了解原因，但解决了修改后端，接口内容不变的问题。

D13 摸了

拆分后端数据返回包，拆分接口。

尝试缓存。

D14灵光乍现，完成了之前放弃的书名功能。评论提取发现大bug

用谷歌api将书名定位到豆瓣详情页，以使用书名搜索功能。

发现好差评评论需要登录的cookie信息才能提取，但是信息每2天就要更新。放弃原本评论提取逻辑。

重开了一个项目文档，**重新写。**

D15 重写豆瓣评论爬虫，打包所有函数，完善错误返回

提取评论的**逻辑重写**了一遍。

错误返回太绕了。一开始不知道要规划，导致后期打布丁非常繁琐，一层层扣，七拼八凑写完总控函数。

D15 写好后端接口

大概了解了django的功能和框架，写好了对应的api

简单使用缓存，异步请求

第一次注册了github并上传后端

D16开始写react

了解react逻辑和需要的结构

知道了手脚架是什么意思

开始建立文件结构

D17稍微弄懂报错数据结构

写前端逻辑，**继续改后端打包的数据结构。**

其实报错那里的数据结构，一直没能懂以全局思考是什么感觉，今天懂了点，在混乱的代码里一通改。有一种终于知道我在写什么的感觉。。。

D18 写了搜索页逻辑

继续了解前端到底怎么写

搜索页逻辑完成并成功运行

D19显示页逻辑

写了一点page2，了解框架。