# Milestone5 Final Report

组名：odometer

组长：3200102642-卢逸凡

组员：3200104866-赵伊蕾 3200104734-沈祺昊 3200102547-梁可愉 3200105119-王柯棣

## 版本迭代过程

我们组项目产品的迭代过程是与需求迭代完全同步的。

课程初始阶段，进行经验以及构建体验的分享，并共同分析进行选题的确定，最终选择 Double-C Analytics Dashboard 进行迭代。

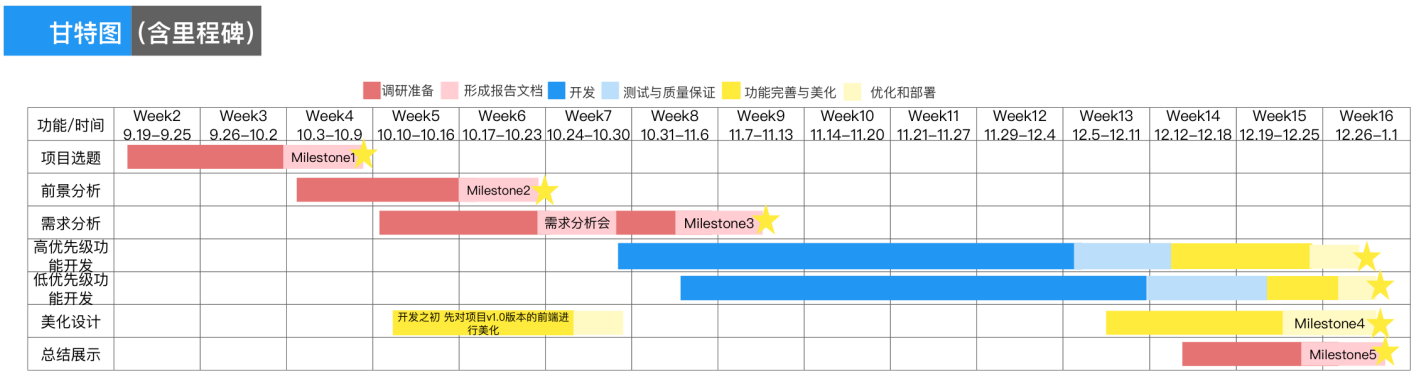
根据助教发布的 Requirements 和 V1.0 版本产品的特点来初步确定项目的需求，进行了专门的线下会议，进一步细化各需求的细节及其实现方式。根据会议确定的需求，进行前景与范围的讨论，同时完成前后端的分工。

根据之前线下会议对于各需求的设想，以及将其扩展详实时所遇见的困惑，以书面文档的方式罗列了我们所要提问的各种问题，设计了访谈会的议程，初步设计了几个 Use Case 在会上讨论，并对于访谈会的结果进行了专门的记录。

组织会议讨论访谈会的结果，获取了比较详实的需求，共同完成 Software Requirements Specification，其中包括六个 Use Case。至此，本项目开发的需求已经初步明晰。

在第一次迭代中，由前后端两组组内独立分工推进，并且在推进的过程中及时提交成果，由对方使用过后提出改进意见，通过钉钉群聊和开发文档实时沟通开发中遇到的困难。开发过程中我们发现 SRS 中的某些需求有些实现起来有困难，有些不够完善，因此我们对需求进行了调整。

在第二次迭代中，我们在会议中对项目表现结果进行评估，发现设计话题数据比较少，此外网页的响应速度不理想。因此我们进一步进行了接口的优化，并补充了一些新的需求，例如增加了非功能需求：提高响应速度，以及低优先级需求：按周统计设计话题分析。同时我们也经过讨论舍弃了一些低优先级需求，例如用户管理系统。




## 后端设计

### 统计Contributor、Issue、Stargazer的Company信息

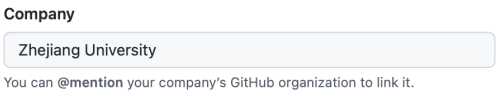
#### 功能

统计Stargazer、Issue、Commit的用户的Company信息，可视化查看项目成员主要来自哪些公司。

#### 算法逻辑

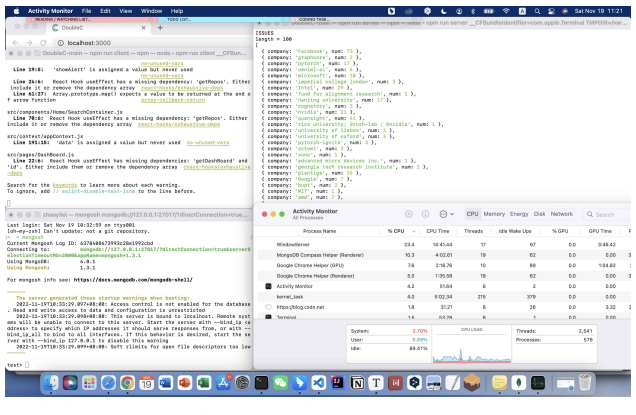
后端数据都是通过Github的Octokit接口获取。由于接口设计问题，用户的Company必须通过2次接口调用才能获取；第一次用以拿到用户列表数据，第二次访问用户接口，拿到company信息。所以在本项目中使用Octokit获取数据所花费的时间比较长。

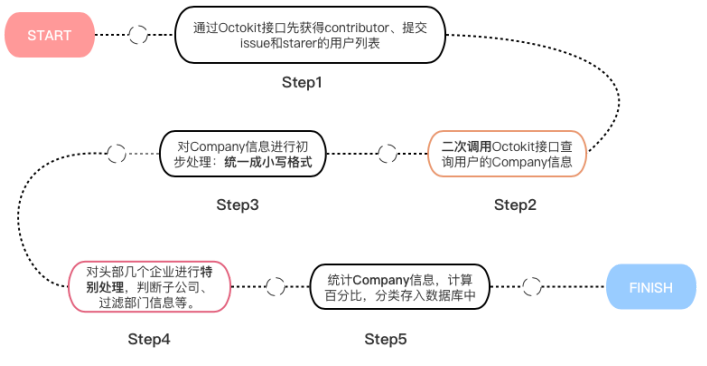
拿到Company初步的信息之后，我们需要在后端对数据进行处理统计之后再存入数据库，以便于前端可以直接获取到各个company的人数和占比，快速渲染界面。Company信息的处理逻辑是比较简单的，首先是删除字段值中的“@”符号，并且统一大小写。公司名称出现“@”字符是因为Github在个人简介处提示，可以通过@来关联自己的公司，所以我们在预处理阶段需要先去除这一字符。




第二步是对不同公司进行合并。有一些开发者会在自己公司名处详细列出自己是公司哪个部门的，所以为了能够更好地统计，我们会去除掉这部分信息。另外，有些大型互联网公司有改名、或者子公司等现象存在，例如meta和Facebook是同一公司，我们要将两个的统计数据进行合并。

左下图是Company这部分调用接口过程中，边处理边输出我们所分类得到的Company信息。右下图是处理流程图。






## 统计Stargazer、Issue、Commit随时间变化的趋势

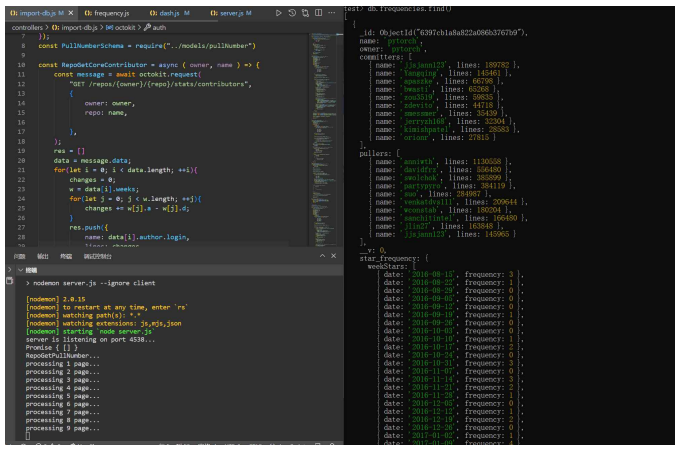
### 功能

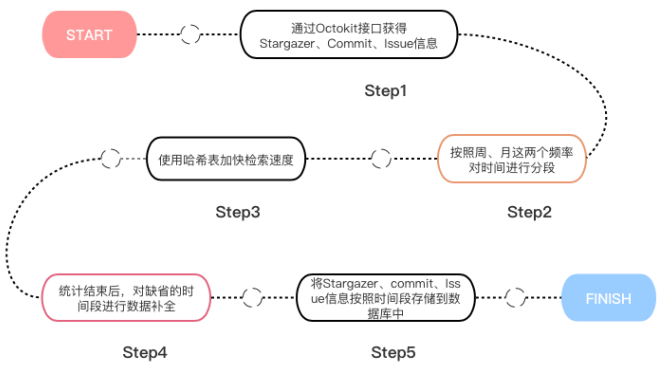
统计Stargazer、Issue、Commit随时间变化的趋势，提供按周统计、按月统计和按年统计三个选择

### 算法逻辑

后端通过Octokit接口获取指定仓库的stargazer、issue、commit的每条记录，根据每条记录的时间统计不同时间尺度的数据频率，使用哈希表加快检索速度。在统计结束后，需要对缺失的时间段进行填充，使整个时间轴是连续的。最后将数据存入数据库。

下图左侧为获取pull request数据的源代码和控制台爬取输出，右侧为爬取完毕的数据库结果。






## 按代码量统计核心用户

### 功能

根据用户的代码量分别统计commit、pull request的核心用户

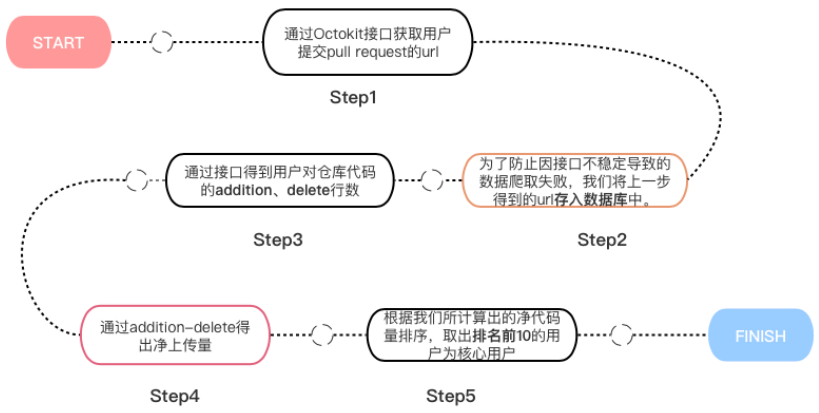
### 算法逻辑

Octokit接口可提供用户addition、deletion的行数，此处使用净增量(addition-deletion)对用户提交代码量进行评估。

统计commit的核心用户可通过接口直接得到。在统计pull request的核心用户中，由于接口限制，必须先获取每条pr的url，再根据url获取该pr的addition和deletion数据。

两次请求严重影响了程序速度，且由于Octokit接口并不稳定，容易出现网络断连、数据缺失的情况，此处对程序接口作出了以下优化：1.首先获取每条pr的url，作为中间结果存储于数据库中；2.从数据库中获取url，此时才真正地遍历url，发送请求并获得数据。

最后，根据代码量进行降序排序。考虑到前端展示的效果，核心用户数的范围为[0,10]。




## 以周/月份为单位统计 pull request 的设计相关数量/关键词

### 功能

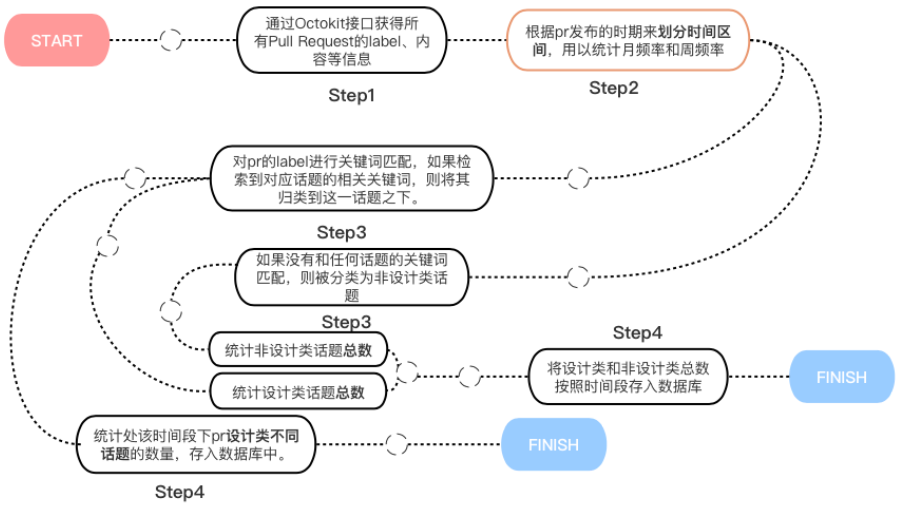
对项目的pull request以周/月为单位进行统计，获得其中设计相关的话题数量，并提取出设计相关的关键词，对各关键词的的数量进行统计。

### 算法逻辑

通过Octokit获取pull request的相关信息，并通过其中的label信息进行统计。对于从接口获取的数据，各pr默认按时间倒序排序。首先逐条取出其详细信息，并从中获取时间信息；若此次时间与上一次统计的时间不同，则对上一个时间区间的数据进行存档，同时将当前区间的统计数据置为零；之后在该时间区间数据的基础上更新数据。

更新数据时，我们对该pull request的label里的设计相关词语进行提取，之后将各词语的数目分别加一即可。

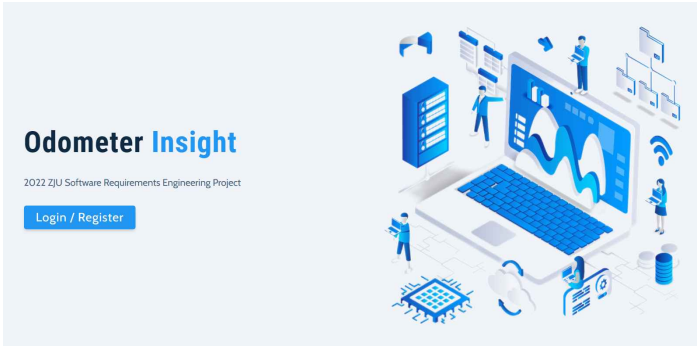
我们将统计的时间范围设定在三年以内，在统计完时将所有数据一次性导入数据库。

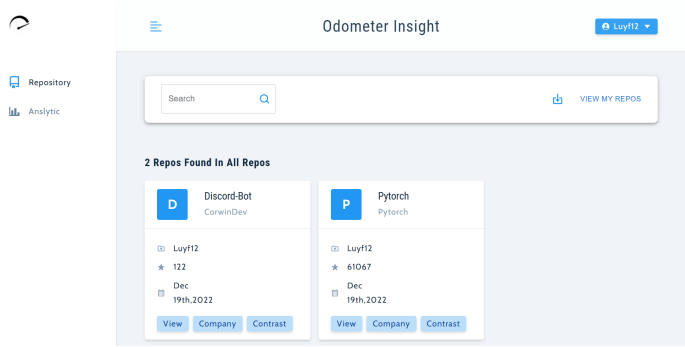



## 前端设计

前端采用react框架搭建，图表绘制采用E-chart，在Double-C基础上新增了三个页面：AnalysisBoard 提供对一个Repo核心用户、发展速度和设计话题的统计分析；CompanyBoard 提供对一个Repo参与人员的公司信息的分析；CompareBoard 提供两个Repo的对比，可以对比的内容包括Repo基本信息和company信息。

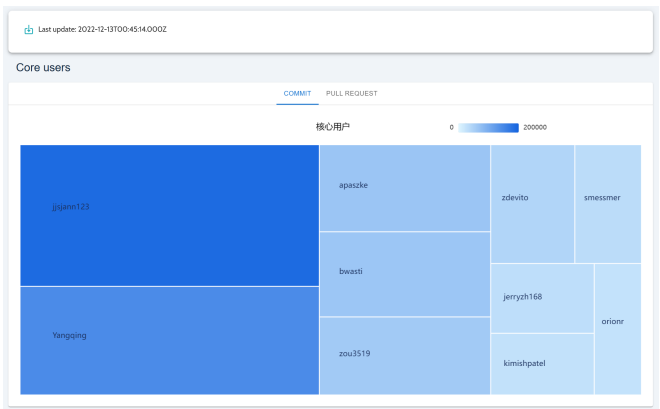
### 页面展示

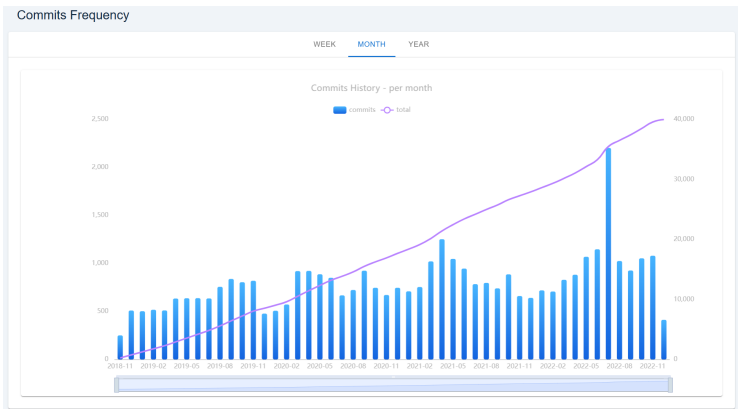


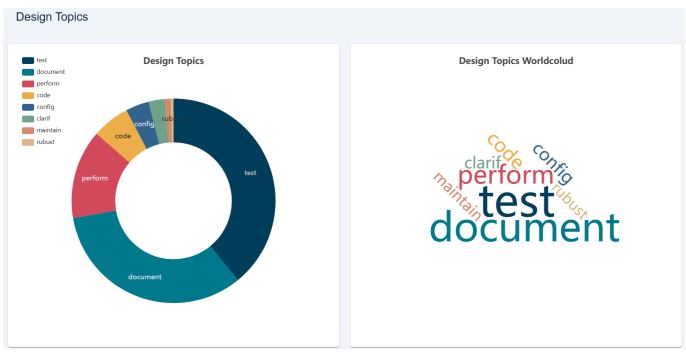



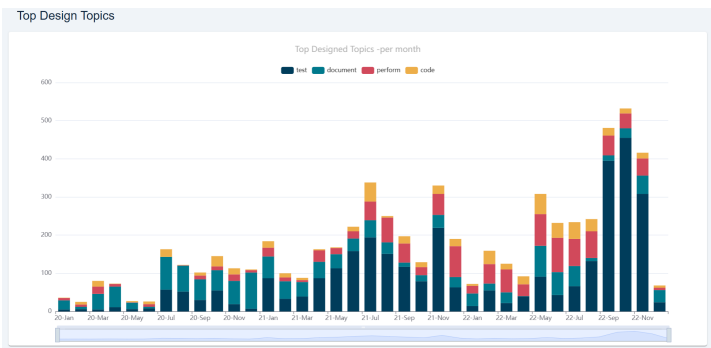

# AnalysisBoard

## 页面展示



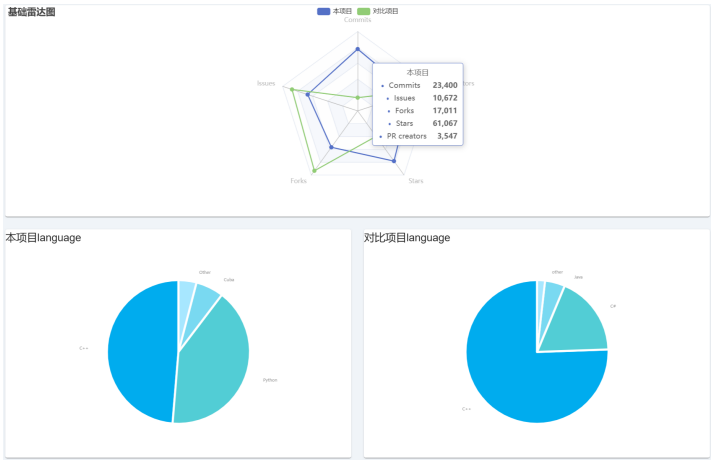

### 功能说明

AnalysisBoard用于分析项目的核心用户、以及社区发展速度。社区发展速度采用stars、issue等参数的变化情况来衡量。

这一部分采用了TAB标签实现数据间隔的切换（周、月、年），TAB标签的作用是在前端页面中可以点击选择想要的页面。比如下图的功能是可以选择（周、月、年）三种不同的时间维度来展现社区发展速度的变化。其中为了实现根据时间维度来展现发展速度这一功能，我们也对从网站上爬取的数据进行了一定程度的处理，根据（周、月、年）统计了star数等信息，最终通过路由将参数传到了前端页面。前端页面用柱形图+折线图展现发展速度，用关系图来展现核心用户，比较美观地将数据进行可视化展示。

# CompareBoard

## 页面展示



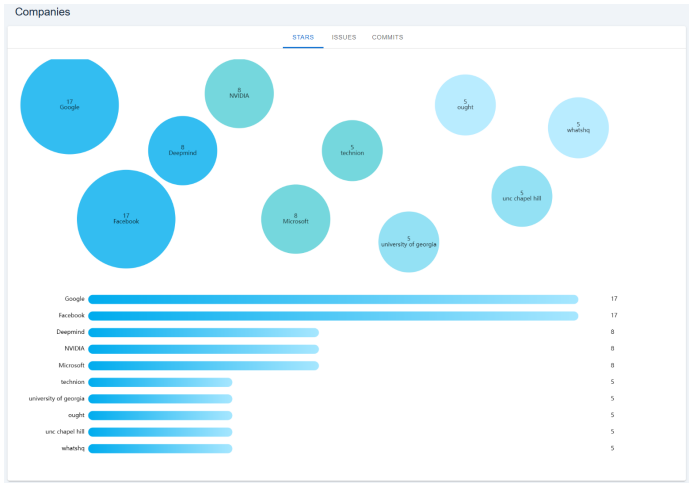


### 功能说明

CompareBoard主要是为了实现项目对比这一功能。因为不同项目的体量差距较大，我们提供了三个与pytorch项目规模接近的项目供用户进行对比。用户可根据自己的需要来对比两个项目的Star数、fork数等信息。同时采用了雷达图比较清晰明了地展现两个项目的基础概况信息。同时也提供了两个项目company信息的对比。

## CompanyBoard

### 页面展示




### 功能说明

CompanyBoard用于展现公司对于项目的贡献情况

利用Tab标签以便于展示star、commit、issue等三个维度下不同公司对该项目的贡献情况。数据利用路由从后端数据库当中获取，格式为{公司名：数目}。前端通过一定的格式处理，统计出了这三个维度下位列前十的公司信息，利用气泡图展现数据的多少，利用排名柱状图来表示前十公司的排名。

# 总结

在本次的课程中，我们通过Double-C的项目改进工作，我们对一款软件的开发流程有了更深一步的认识。在过往的认知中，软件开发的流程就只是拿到要求，接着一头雾水地硬干下去，完全没有需求分析地意识；而通过这次的课程，我们不仅认识到了需求的重要性，同时还了解到了需求分析的流程与要素，掌握了需求分析的工具与方法，尤其是对UML这一重要的工具进行了深层次的掌握。

更重要的是，除开这些理论之外，还有实际的项目开发供我们对知识进行实践层面的认知。在实际的项目开发中，我们切身体会了一个项目从无到有的过程，得到了由学业界到工业界的过渡机会，切实地体验了由项目分析，到Vision & Scope的确定，再到后来的SRS书写、编码完成，并在此过程中不断完成需求迭代，将我们的项目推向更好。这个项目对我们来说，也是一笔很宝贵的财富。