软件需求工程 实验一：需求获取

实验报告

一、组员及成绩分配

学号	姓名	成绩分配比例
181860031	胡欣然	0.34
181860058	刘雅婷	0.33
181860138	张问津	0.33

二、实验目的

选择较为熟悉的知名 IDE：VSCode 作为分析对象，在 GitHub 中 VSCode 项目下的 Issue[1]中进行数据获取，通过需求分析和分类，获取用户对 VSCode 的潜在需求。

三、实验方法及过程

（一）需求获取

需求获取共分为两部分，一部分是对 closed 状态 Issue 的获取，另一部分是对 open 状态 Issue 的获取。二者在代码中的差别仅在于对应 url 中 state 值和保存 Excel 文件时命名的不同。下面以 open 状态 Issue 的获取[2]为例。

总体步骤包括：

1. 输入要获取的 Issue 页码范围；
2. 读取 Issue 信息到字典 issue_dict 中；
3. 将获取的信息存储到 Excel 工作表中。

下面是具体步骤：

1. 输入要获取的 Issue 的页码范围

   需要通过 input 语句输入开始页码 start 和结束页码 end。

2. 读取 Issue 信息到 issue_dict 中

首先根据当前准备获取的 Issue 页码 i，通过 GitHub 提供的 API 接口设定好 url（page={i}），再在自定义的 read_json 函数中获取 url 对应的 Issue 信息并存储到 issue_dict 字典中。

```
for i in range(int(start),int(end)):
    url=f"https://api.github.com/repos/microsoft/vscode/issues?state=open&page={i}"
    read_json(url)
    print('processing %d out od %d items...'%(i+1,int(end)),'\r',end='')
```

1) issue_dict 字典：

issue_dict = { key:value}　key:issue title, value:class Repositories

Repositories 类的数据成员包括：number, title, status, create_time, closed_time, body。

```
class Repositories:
    def __init__(self,number,title,status,create_time,closed_time,body):
        self.issue_number = number
        self.issues_title = title
        self.issues_status = status
        self.create_time = create_time
        self.closed_time = closed_time
        self.label_name_list = []            #Labels may have a lot
        self.issue_body = body
```

2) read_json(url)函数：

先调用自定义的 get_issue_json(url)函数，通过 r=request.get(url) 构造一个向服务器请求资源的 url 对象。这时候的 r 是一个包含服务器资源的 Response 对象，包含从服务器返回的所有的相关资源。然后再返回结果的 JSON 对象。

```
def get_issue_json(url):
    r=requests.get(url)
    return r.json()
```

再利用获得的 JSON 对象构造相对应的 Repositories 对象并插入 issue_dict 字典中。

```
def read_json(url):
    file = get_issue_json(url)
    for i in file:
        issue_dict[i["title"]] = Repositories(\
            i["number"],i["title"],i["state"],i["created_at"],i["closed_at"],i['body'])

        # append label name to label list
        for j in i["labels"]:
            issue_dict[i["title"]].label_name_list.append(j["name"])
```

3. 将获取的信息存储到 Excel 工作表中

利用 Python 中的 openpyxl 库[3]，初始化一个 workbook、将工作表激活并插入列名后，通过调用自定义的 write07Excel 函数，读取 issue_dict 中的元素，将其插入表格中并保存。

```
def write07Excel(path,value,sheet):
    sheet.append(value.result_xl())
    wb.save(path)
```

```
wb = openpyxl.Workbook()
sheet = wb.active
sheet.append(["issue_body","issue_number",\
              "issues_title","issues_status","create_time","closed_time","label_name_list"])
for i in issue_dict.values():
    write07Excel(f"open_{start}_{end}.xlsx",i,sheet)
```

（二）需求分析

1、词云分析

“词云”就是通过形成“关键词云层”或“关键词渲染”，对网络文本中出现频率较高的“关键词”的视觉上的突出。

词云图过滤掉大量的文本信息，使浏览网页者只要一眼扫过文本就可以领略文本的主旨。

我们采用这种方法，过滤掉无意义的信息，提取高频关键词。由关键词回溯获得的需求数据，更高效地获得了有意义的需求案例，便于分析。

由需求获取阶段获得的数据，抽取“title”一栏进行词云分析[4]。

首次分析，图片中出现了很多诸如 in，and 之类的无意义高频词，于是在代码中添加忽略这些词汇的逻辑，并重新分析。

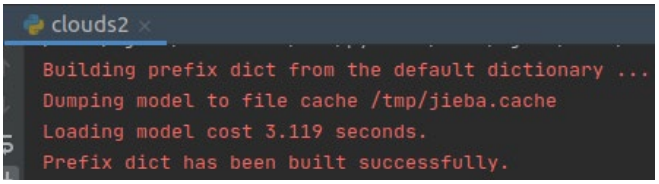
```
stopword=['in', 'to', 'the', 'not', 'on', 'is', 'when', 'for', 'with', 'and', 'of', 'vscode', 'eroor']
stopword=stopword+ls
```

图片 1 词云分析时忽略无关词汇

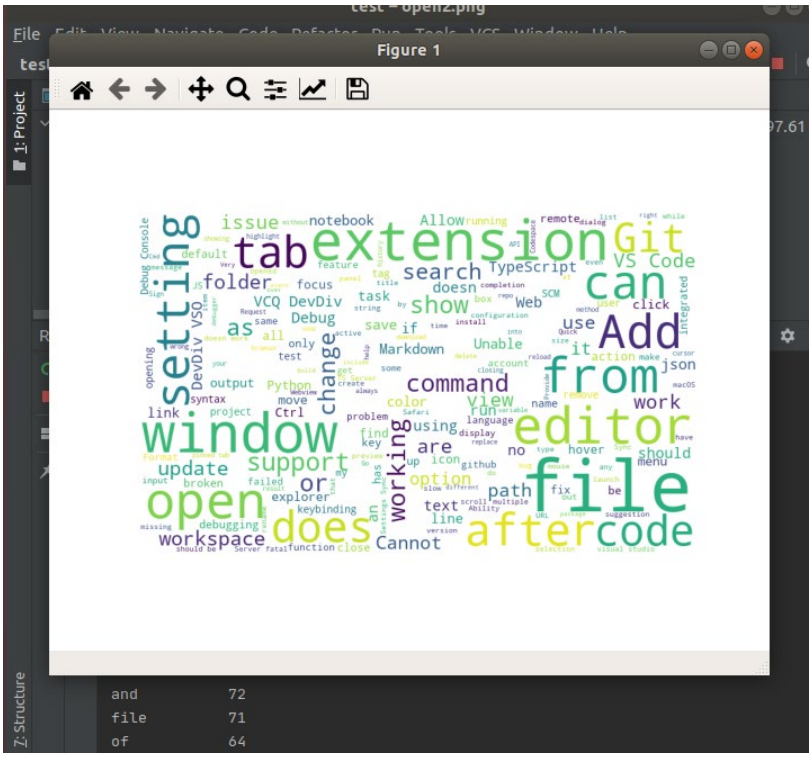
总体步骤如下：

1) 读取文本文件
2) 进行文本切割
3) 统计词频
4) 按照频率对单词排序
5) 去掉无意义常用词
6) 打开底图
7) 获取底图信息，转化为数组
8) 绘制图片
9) 生成图片

生成结果如下（图片附在实验项目中）：



图片 2 词云分析输出



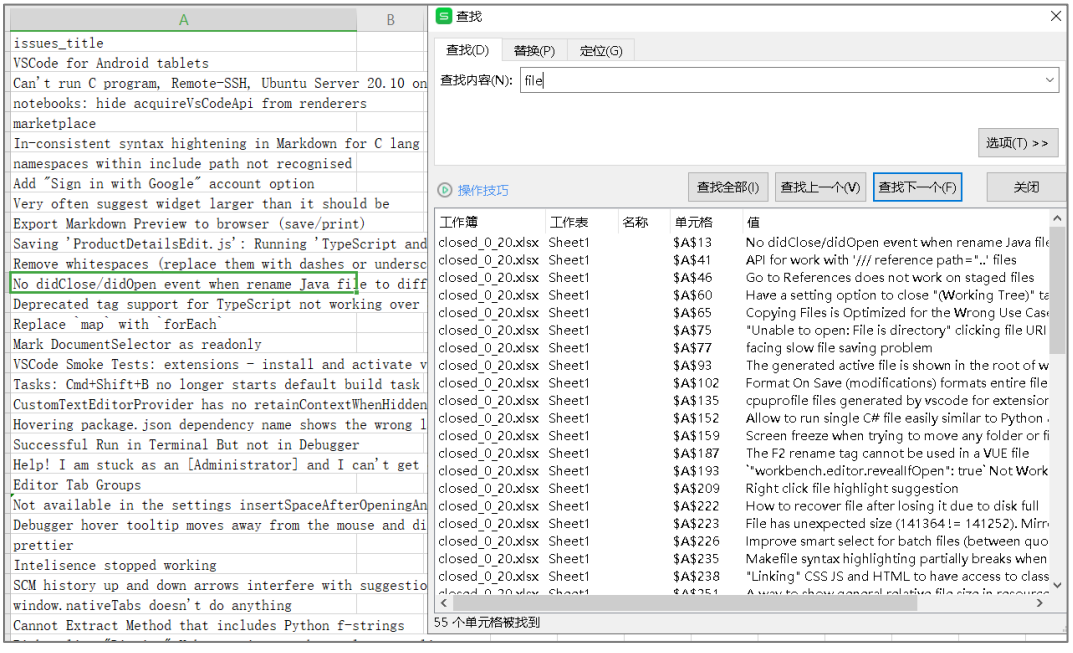
图片 3 词云分析结果

2、关键词回溯问答

从得到的词云图片中，选取了六个出现频繁且比较具有代表性的词：

“file” “terminal” “editor” “workspace” “extension” “git”

分别在得到的原始数据中搜索涉及这六个词的问答，以 file 为例：



图片 4 回溯关键词“file”

1）在 title 栏中检索“file”一词，得到 55 条搜索结果。

2）阅读 55 条搜索结果，按照标题信息可得到的内容，筛选出 8 条可以提取出需求的问答：

1. *No didClose/didOpen event when rename Java file to different cases*
2. *Go to References does not work on staged files*
3. *Have a setting option to close "(Working Tree)" tab when pressing "Open File"*
4. *Copying Files is Optimized for the Wrong Use Case (Very, Very Rarely Does Anyone Want "Foo (Copy).js")*
5. *Avoid the reload required for each change in the snippets.json file*
6. *Source Control file list is super hard to read*
7. *To be able to accept all current/incoming changes when solving merge conflicts in a file*
8. *File is not active in tab when opened from search results*

3、从问答中获取需求

继续以“file”为例，比如第 6 条，

6. Source Control file list is super hard to read

可以看出，问答内容是源代码管理文件列表可读性差，那么从这条中提取的需求就是：改进源代码管理文件列表，使其有较好的可读性。

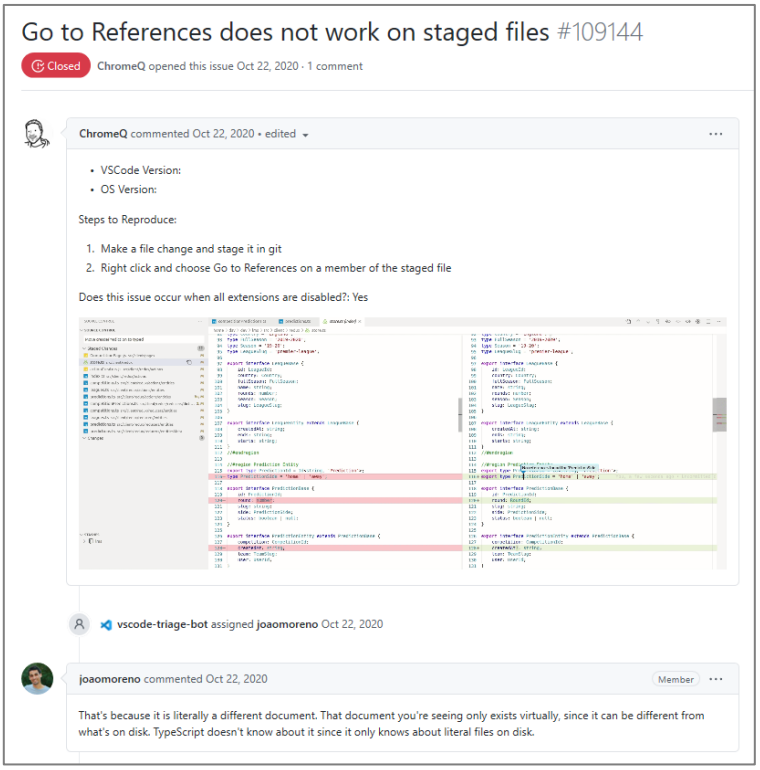
部分问答不能直接从标题看出需求，则回到该条问答的 GitHub issue，查看具体的内容。

例如第 2 条，

2.Go to References does not work on staged files

查到原有的#109144 问答，可以得知，“*Go to References*”功能不可以正常运作在暂存文件上，下有的回答解释了这一问题，并且表明这并不是一个已经被解决的问题。所以得到一条潜在的需求：

暂存文件应当可以使用“*Go to References*”功能。



图片 5 查看对应的 Issue 并分析需求

（三）需求分类

类似地，我们得出了 53 条可用需求：

从Issue到需求（未分类）

1. #109117：Editor tab shadow is rendered on top of contrast border

   需求：IDE的GUI界面应当简洁美观、操作流畅，开发人员应当修复GUI界面内部显示bug。

2. #109060：Without any editors open the find widget doesn't show in the Output pane

   需求：在未打开任何编辑器情况下，在输出栏中也应该显示查询窗口，以保证用户在未打开编辑器时也能搜索关键词。

3. #108868：onDidChangeActiveTextEditor fired twice when closing the editor

   需求：在用户使用快捷键关闭某一个编辑器窗口时，`onDidChangeActiveTextEditor`只应当被触发一次。

图片 6 未分类的 issue 以及对应需求（节选）

完整需求见项目文档：**未分类的 issue 以及对应需求.md**

按照内容和性质，需求可分为外部接口需求、功能需求、其他非功能需求这三大类：

1） 外部接口需求

1.1 用户界面需求

2） 功能需求

2.1 对 IDE 编辑器的需求

2.2 对 IDE 终端的需求

2.3 对 IDE 控件、工具的需求

2.4 对 IDE 插件的需求

2.5 对 IDE 中 Git 使用的需求

2.6 对 IDE 处理文件的需求

3） 其他非功能需求

3.1 性能需求

3.2 易用性

3.3 其他需求

四、实验结果和效果分析

（一）实验结果

最终的需求文档如下：

1 外部接口需求

1.1 用户界面

1） 在插件选项卡中，应当减小插件标题占用的垂直空间。在已打开底部终端的情况下，插件标题占用大量垂直空间会导致显示插件内容的空间很少。

2） 用户可以选择使用终端、调试控制台、问题和输出的浮动窗口，以减少用户在不同窗口间来回切换的操作，方便用户利用大屏幕空间或者多个显示器。

3） 在使用编辑器时，用户可以对标签进行分组管理。

2 功能需求

2.1 对 IDE 编辑器的需求

1） 对于涉及 editor 的操作，可以实现撤销和恢复。

2） 用户可以自定义编辑器的配置文件。

3）编辑器对于接受输入建议应当默认为关闭，不自动弹出。

2.2 对 IDE 终端的需求

1）应当支持用户使用多个终端的选项卡，而不是默认的单个终端。

2）应当允许插件修改终端的环境变量，以方便用户在项目本地设置 SDK 路径。

3）在启动终端之前，用户可以通过设置一个选项来设置环境变量。

4）用户可以在终端中搜索所有文件。

5）用户能够通过主菜单中的选项来终止当前终端。

6）终端应当支持快速滚动，用户在按住 alt 建的同时滚动鼠标滚轮，可以通过 1 次鼠标滚轮滚动在终端中滚动 1000 行。

7）应当在集成终端面板中添加“clear”按钮，用来清空终端的内容。当前 IDE 的集成终端面板只有“add”和“kill”两个按钮，用户无法通过界面按钮来清空终端内容。

8）用户可以通过插件获取和控制终端的滚动位置。

9）用户可以通过鼠标滚动对集成终端进行缩放，以方便使用较小屏幕的用户使用集成终端。

2.3 对 IDE 控件、工具的需求

1）在搜索功能中，搜索结果可以按照字段进行排序。

2）更改设置时，可以在点击"Open File"时更改"(Working Tree)" 状态。

3）终端应当可以跨工作区重复使用。

4）在 telemetry 功能关闭时，也应当能在根目录中使用工作区。

5）在使用工作区的搜索功能时，应当能忽略部分文件夹。

6）工作区可以保留选定的源代码控制仓库。

7）工作区添加或删除文件夹时，不应当重新加载窗口。

8）用户应当能够使用自定义编辑器创建文件，而无需具有打开的工作区。

2.4 对 IDE 插件的需求

1）插件能够将源文件一些元数据添加到文档中，用户使用快捷键运行程序时，IDE 基于元数据执行不同的行为。例如，插件判断源文件中是否有 main()方法，若有则运行该应用程序；插件判断源文件中是否有测试用例，若有则运行测试用例。

2）使用.NET CORE 测试调试时，IDE 对于插件的路径名的最大值限制应当允许用户自行设定，否则会出现插件的路径名过长导致无法调试的现象。

3）在安装拓展程序时，确认安装的通知对话框中应当显示安装的插件列表。

4）在插件安装或更新失败时，应当提供一个“显示详细信息”的选项。

5）对于没有安装的插件，应当提供一个筛选器，以便用户查询特定类型的未安装的插件。

6）IDE 应当为插件添加“教育”的类别。

7）IDE 应当允许用户禁止接收插件的更新通知，以便用户在愿意使用旧版本插件的情况下不被更新通知打扰。

8）IDE 应当允许用户自行对插件进行分组，以方便用户管理大量插件。

9）添加收藏插件的功能，以便用户在发现感兴趣的插件但是没有安装的情况下能够再次找到这个插件。

2.5 对 IDE 中 Git 使用的需求

1）应当添加 Git 缓存删除器，以便开发过大量使用 Git 的项目的用户清理 Git 缓存空间。

2）在通过插件改变 IDE 的语言的情况下，其 Git 模块应当仍然使用英语，否则会出现一些关于 Git 的名词的不妥当的翻译。

3）IDE 应当支持使用命令行删远程 Git 标签。当前 IDE 只支持删除本地标签。

4）用户在 git clone 项目时，应当可以指定目录名称。

5）用户在远程跟踪分支的情况下，应当禁用创建分支的选项。

6）用户 git commit 失败时，应当显示完整的 git commit 的输出信息。

7）如果用户在 git clone 之后没有打开任何文件夹，应当提供一个选项，用户通过选择该选项来自动打开 git clone 的文件夹。

8）在 Git 输出控制台中，应当显示 Git 进度的详细信息。

9）IDE 应当支持多个作者共同对 git commit 信息进行署名。

10） git diff 编辑器应当在描述信息中显示资源的路径。

2.6 对 IDE 处理文件的需求

1）暂存文件应当可以使用“ go to references ”功能。

2）应当避免每次更改 snippets.json 文件时都需要重新加载。

3）解决文件中的合并冲突时，应当能够接受所有 current/incoming 状态的更改。

4）在搜索结果中打开文件时，文件应当处于活动状态。

3 其他非功能需求

3.1 性能需求

1）在点击 Github/VSCode 图标时，不应当重新加载编辑器，以减少用户的等待时间。

3.2 易用性

1）IDE 的 GUI 界面应当简洁美观、操作流畅，开发人员应当修复 GUI 界面内部显示 bug。

2）在未打开任何编辑器情况下，在输出栏中也应该显示查询窗口，以保证用户在未打开编辑器时也能搜索关键词。

3）在用户使用快捷键关闭某一个编辑器窗口时，onDidChangeActiveTextEditor 只应当被触发一次。

4）在对 Java 文件重命名时，应当分别对原来 URI 和新的 URI 触发 didClose 和 didOpen 事件。

5）用户生成的活动文件不应当被放置在工作区的根目录中，以便用户对于工作区进行管理。

3.3 其他需求

1）IDE 的源代码文件可读性差，应当进行改进其可读性。

（二）效果分析

本次实验对于 Visual Studio Code 在 GitHub 上的 Issue 进行了分析，筛选出 6 个高频关键词（editor，terminal，workspace，extension，Git，file），针对每个关键词筛选出约 10 条典型的 Issue 并从中提取出需求，最后对于提取出来的共 53 条需求进行分类。

这些需求大体上被分为外部接口需求、功能需求和其他非功能需求三大类。

在外部接口需求这一类别中，包含关于用户界面的需求。一些 Issue 涉及了对于 VSCode 的编辑器、集成终端、插件选项卡等界面的问题反馈。例如，小屏幕用户认为插件选项卡的标题所占空间太大、导致插件内容版块所占空间较小；大屏幕或多显示器用户认为可以通过浮动窗口增加屏幕的利用率。从这些关于界面的 Issue 中，提取出了 3 条针对用户界面的需求。

从 Issue 中提取出的功能需求是数量最多的一类。这里，按照这些需求涉及的关键词分为了 6 个类别，分别是关于编辑器、终端、控件与工具、插件、Git、文件的需求，这些需求指出了 VSCode 应当添加的功能。这些需求来自于用户提出几十条 Issue，用户通过这些 Issue 表达了在使用 VSCode 的过程中尚且不太方便的地方，并且认为增加某一功能后会有所改进。

在其他非功能需求这一类别中，包括性能需求、易用性和其他需求三部分。

在性能需求的部分，包括减少用户等待时间的需求；在易用性的部分，包括消除界面显示 bug、增大某些控件的使用范围等需求；在其他需求的部分，包括提高 IDE 源代码可读性、便利共同迭代开发的需求。

总体上来说，从 Issue 中提取的需求会比较琐碎。这些需求有可能会涉及一些细节问题，只与一小部分用户相关，或者属于特定条件下的需求。例如，“在对 Java 文件重命名时，应当分别对原来 URI 和新的 URI 触发 didClose 和 didOpen 事件”这一需求，只与使用 Java 语言进行开发的用户相关，甚至于一部分使用 Java 语言的用户也不关心这一需求；“使用.NET CORE 测试调试时，IDE 对于插件的路径名的最大值限制应当允许用户自行设定”这一需求，只被使用.NET CORE 测试进行调试并且插件的路径名过长的一小部分用户关心，其他用户不会遇到由于插件的路径名过长导致无法调试的现象；“在通过插件改变 IDE 的语言的情况下，其 Git 模块应当仍然使用英语”这一需求，只与通过安装插件改变 VSCode 的系统语言且该插件将与 Git 相关的专有名词进行了不恰当的翻译的一部分用户有关，这部分用户希望通过非英语语言使用 VSCode，但是不希望 Git 等专有名词也被一同翻译。上面举例的几个需求的确是某一部分用户在特定情况下所关心的需求，但是当它们被写入一份仅包含五十余条需求的需求文档时，将会稍显琐碎、不成体系。

其次，VSCode 作为一个支持插件安装的软件，用户使用过程中的一些问题其实是与某些插件相关的。因此，从 Issue 中提取出来的一些需求实际上与某些特定的插件有关，例如上文中提及的路径名称过长导致无法调试的插件。这些需求是出于 VSCode 本身的问题，而不是插件的问题，但是这些问题仅仅在使用特定插件时才会产生。这使得通过 Issue 整理出来的需求涉及到一些琐碎的细节问题。

另外，从 Issue 分析需求的过程中，有可能会遇到一些比较有争议的需求。例如，Issue 的发布者认为应当增添某个功能，但是其他开发者在该 Issue 下回复表示这个功能在带来一些便利的同时也会引发一些逻辑上的冲突。

VSCode 作为一个大型的、比较成熟的开源项目，当前已经有众多活跃用户，也有许多开发者在开源社区中积极提交 Pull Request 并完成 Merge、共同完善这个项目。因此，这个项目当前仍然存在的问题很有可能是比较琐碎的问题，只与一小部分用户相关，只有在特定情境下才能重现，或者具有争议；这一特点也反映在该项目的 Issue 中。所以，当我们通过 Issue 提取需求时，这些需求有可能比较琐碎或者具有争议，加上我们详细分析的 Issue 只有六十条左右，这也使得最终分类后的需求的体系尚且不是非常完整。

五、结论

在保证准确度和技术水平有限的情况下，我们小规模地从 GitHub 上获取了一些关于 VSCode 的问答。对问答的标题进行了词云分析，并从中抽取关键词进行聚类。

如实验方法中描述，我们抽取了 50 余条需求，对其进行回溯。在具体的 Issue 里提取出该问答潜藏的需求。

经过分析，将这些需求分为三大类：外部接口需求、功能需求和其他非功能需求。

其中有较少的外部接口需求，即用户界面需求。最多的是功能需求。按照这些需求涉及的关键词分为 6 个类别，分别是关于编辑器、终端、控件与工具、插件、Git、文件的需求。这些需求大多是用户希望 VSCode 添加或改进的功能。

最后一类是非功能需求。包括性能需求、易用性和其他需求三部分。这些需求中包括 IDE 本身存在的、需要修复的 bug。也包括用户希望缩短响应时间、提高易用性等需求。

由于信息源的庞杂，以及 VSCode 本身是一个存在时间很长，功能繁多的开源项目。提取出的需求较琐碎，在人工提取过程中，我们尽力使其具有一定的代表性，但无法避免某一些问题不可复现，是个别用户的特殊问题。并且，数据量的大小决定我们不能形成一份很全面的文档。详情已在效果分析中说明。

总的来说，本次实验，我们在实践中学习了如何评估并选择信息源、通过爬虫获取信息、由代码进行词云分析、在 GitHub 的 Issue 中提取有意义的需求，对获取的需求进行分类.....选择了 VSCode 作为我们要探索的开源项目，并获得了一定的有效信息。在此过程中实践了需求的获取、分析和分类。

VSCode 作为一个庞大而复杂的开源项目，用户群体巨大，需求全面多样，并且具有很明显的各异性。

由于使用环境、使用目的的差别，用户提出的需求也有差别。在已有的 VSCode 功能里，较有代表性或某一群体的普遍需求已经大部分被解决。现有的新需求多为个别用户的个别需求，或者某一类新增后会与原有功能产生矛盾的需求，或实现代价很大且实现价值较小的需求。

参考文献：

[1] GitHub VSCode 项目 Issue 来源 https://github.com/microsoft/vscode/issues

[2] GitHub Issue 爬取 https://github.com/Tc-blip/Github_issue_spyder

[3] openpyxl 学习笔记 https://blog.csdn.net/u010916953/article/details/51447993

[4] Python 词云分析 https://www.cnblogs.com/wydxry/p/10224426.html