**基于社区化网络的**

**视频动态团购APP**

**——项目需求工程**

配

置

文

档

管

理

课 程： 软件需求分析原理与实践

题 目：基于社区化网络的视频动态团购APP项目需求工程

专 业： 软件工程

班 级： 1802

项目小组： G05小组

说明：

l.《配置管理文档》(CM)是项目初期策划的结果，它提供了小组项目的流程以及一个统一的操作大纲，方便进行配置管理，保留项目的进程以及里程碑

2.本文档版本演进

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **版本** | **修订日期** | **修订人** | **修订说明** | **修订状态** | **审批日期** | **审核人** | **批准人** |
| 0.1.0 | 2021-03-27 | 黄德煜 | 首次创建 | S | 2021-03-27 | 朱涵 | 邢海粟 |
| 0.1.1 | 2021-04-01 | 朱涵 | 修改 | M |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**修订状态：S--首次编写，A--增加，M--修改，D--删除；**

**日期格式：YYYY-MM-DD。**

目录

[1引言 3](#_Toc67945746)

[1.1标识 3](#_Toc67945747)

[1.2背景 3](#_Toc67945748)

[1.3项目概述 3](#_Toc67945749)

[1.4文档概述 4](#_Toc67945750)

[2版本号管理 4](#_Toc67945751)

[2.1版本号 4](#_Toc67945752)

[2.2版本控制规范 4](#_Toc67945753)

[3 github配置管理 6](#_Toc67945754)

[3.1配置界面 6](#_Toc67945755)

[3.2管理流程 7](#_Toc67945756)

[3.2.1 创建分支git操作（以git desktop为案例） 7](#_Toc67945757)

[3.2.2 管理分支git操作 8](#_Toc67945758)

[3.2.3 合并分支git操作 8](#_Toc67945759)

# 1引言

## 1.1标识

标题：《配置管理文档》

简称：CM

版本号：0.1.0

## 1.2背景

1. 待开发软件系统的名称：基于社区化网络的视频动态团购APP
2. 本项目的任务提出者：杨枨老师以及企业助教
3. 开发者：浙大城市学院SRA-2021-G05小组
4. 用户：杨枨老师以及企业助教
5. 实现该软件的计算中心或计算机网络：阿里云

## 1.3项目概述

1.项目用途

基于社区定位的，以视频形式分享为主的团购APP

1. 项目特性

(1).主功能：打开界面后进行即时发布动态或者进行直播，通过输入关键字检索商家团购产品及优惠，以团长的名义发布动态或者直播的形式发布在附近的人群当中，根据最终拼团的人数多少得到不同的优惠额度，团购结束，团长得到佣金，团员获得商品优惠价。

(2).进阶功能：团长及商家评价、社区聊天功能。

3.项目投资方

SRA2021-G05小组

4.项目需方、用户

杨枨老师、浙大城市学院在校学生、企业助教

5.项目开发方

SRA2021-G05小组

6.支持机构

浙大城市学院计算机与计算科学学院

1. 用户代表

杨枨老师、浙江大学城市学院在校学生、企业助教

## 1.4文档概述

本文档主要针对开发人员和项目经理编写的。配置管理文档的目的就是能够统一配置的管理，帮助项目在前进的路上不走歪路，留有里程碑。

# 2版本号管理

## 2.1版本号

代码版本格式：主版本号.次版本号.修订号，版本号递增规则如下：

1. 主版本号：当你做了不兼容的 API 修改，

2. 次版本号：当你做了向下兼容的功能性新增，

3. 修订号：当你做了向下兼容的问题修正。

报告（PPT）版本格式：正式版本号.评审版本号.自行迭代号，版本号递增规则如下：

1. 正式版本号：正式评审过后的正式版本，一般为1.0.0，提交日期为评审当周的周日，

2. 评审版本号：经过预评审或先行组评审后，根据其评审内容修改的版本，

3. 自行迭代号：根据上课内容或小组会议后，对上一版本内容的修正。

先行版本号及版本编译元数据可以加到“主版本号.次版本号.修订号”的后面，作为延伸。

## 2.2版本控制规范

以下关键词 MUST、MUST NOT、REQUIRED、SHALL、SHALL NOT、SHOULD、SHOULD NOT、 RECOMMENDED、MAY、OPTIONAL 依照 RFC 2119 的叙述解读。（译注：为了保持语句顺畅， 以下文件遇到的关键词将依照整句语义进行翻译，在此先不进行个别翻译。）

1. 使用语义化版本控制的软件必须（MUST）定义公共 API。该 API 可以在代码中被定义或出现于严谨的文件内。无论何种形式都应该力求精确且完整。

2. 标准的版本号必须（MUST）采用 X.Y.Z 的格式，其中 X、Y 和 Z 为非负的整数，且禁止（MUST NOT）在数字前方补零。X 是主版本号、Y 是次版本号、而 Z 为修订号。每个元素必须（MUST）以数值来递增。例如：1.9.1 -> 1.10.0 -> 1.11.0。

3. 标记版本号的软件发行后，禁止（MUST NOT）改变该版本软件的内容。任何修改都必须（MUST）以新版本发行。

4. 主版本号为零（0.y.z）的软件处于开发初始阶段，一切都可能随时被改变。这样的公共 API 不应该被视为稳定版。

5. 1.0.0 的版本号用于界定公共 API 的形成。这一版本之后所有的版本号更新都基于公共 API 及其修改内容。

6. 修订号 Z（x.y.Z | x > 0）必须（MUST）在只做了向下兼容的修正时才递增。这里的修正指的是针对不正确结果而进行的内部修改。

7. 次版本号 Y（x.Y.z | x > 0）必须（MUST）在有向下兼容的新功能出现时递增。在任何公共 API 的功能被标记为弃用时也必须（MUST）递增。也可以（MAY）在内部程序有大量新功能或改进被加入时递增，其中可以（MAY）包括修订级别的改变。每当次版本号递增时，修订号必须（MUST）归零。

8. 主版本号 X（X.y.z | X > 0）必须（MUST）在有任何不兼容的修改被加入公共 API 时递增。其中可以（MAY）包括次版本号及修订级别的改变。每当主版本号递增时，次版本号和修订号必须（MUST）归零。

9. 先行版本号可以（MAY）被标注在修订版之后，先加上一个连接号再加上一连串以句点分隔的标识符来修饰。标识符必须（MUST）由 ASCII 字母数字和连接号 [0-9A-Za-z-] 组成，且禁止（MUST NOT）留白。数字型的标识符禁止（MUST NOT）在前方补零。先行版的优先级低于相关联的标准版本。被标上先行版本号则表示这个版本并非稳定而且可能无法满足预期的兼容性需求。范例：1.0.0-alpha、1.0.0-alpha.1、1.0.0-0.3.7、1.0.0-x.7.z.92。

10. 版本编译元数据可以（MAY）被标注在修订版或先行版本号之后，先加上一个加号再加上一连串以句点分隔的标识符来修饰。标识符必须（MUST）由 ASCII 字母数字和连接号 [0-9A-Za-z-] 组成，且禁止（MUST NOT）留白。当判断版本的优先层级时，版本编译元数据可（SHOULD）被忽略。因此当两个版本只有在版本编译元数据有差别时，属于相同的优先层级。范例：1.0.0-alpha+001、1.0.0+20130313144700、1.0.0-beta+exp.sha.5114f85。

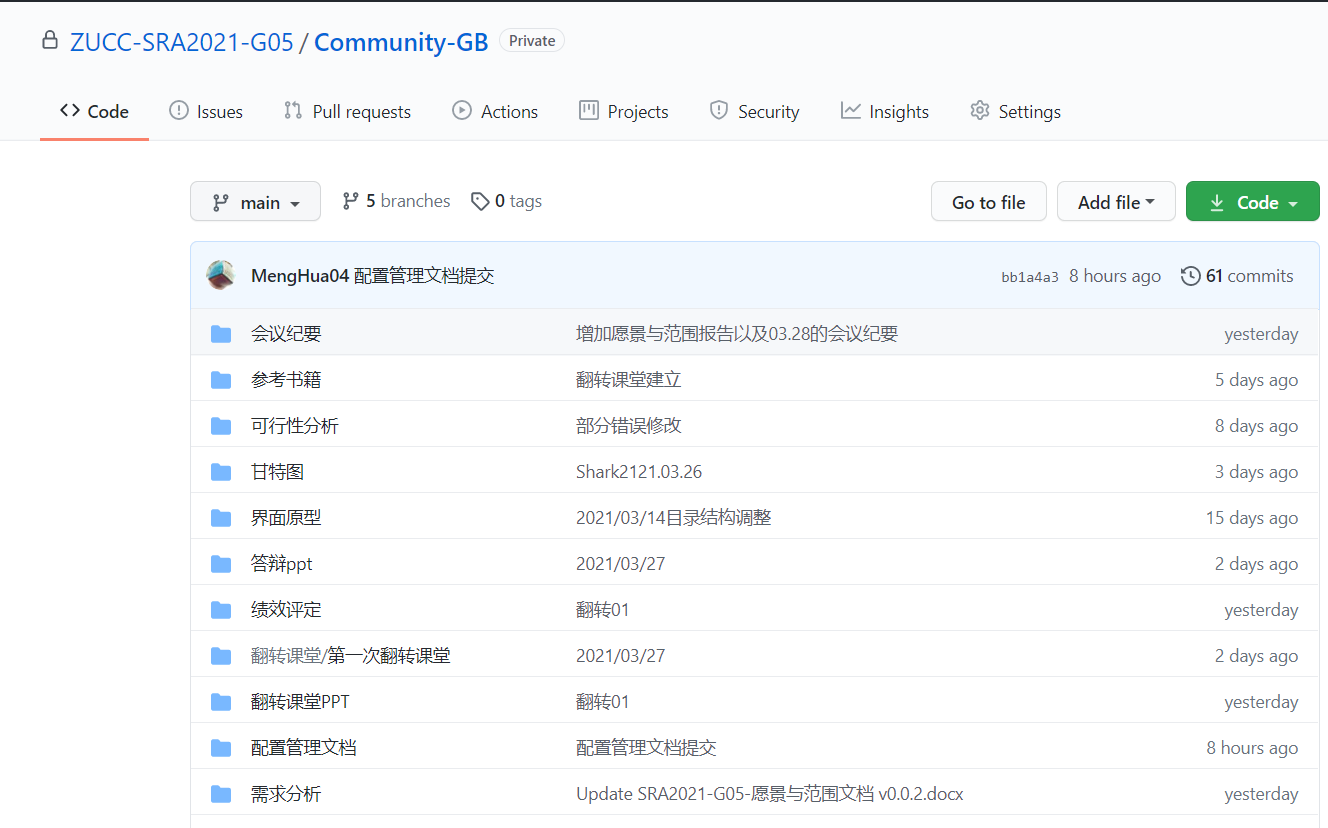
11. 版本的优先层级指的是不同版本在排序时如何比较。判断优先层级时，必须（MUST）把版本依序拆分为主版本号、次版本号、修订号及先行版本号后进行比较（版本编译元数据不在这份比较的列表中）。由左到右依序比较每个标识符，第一个差异值用来决定优先层级：主版本号、次版本号及修订号以数值比较，例如：1.0.0 < 2.0.0 < 2.1.0 < 2.1.1。当主版本号、次版本号及修订号都相同时，改以优先层级比较低的先行版本号决定。例如：1.0.0-alpha < 1.0.0。有相同主版本号、次版本号及修订号的两个先行版本号，其优先层级必须（MUST）透过由左到右的每个被句点分隔的标识符来比较，直到找到一个差异值后决定：只有数字的标识符以数值高低比较，有字母或连接号时则逐字以 ASCII 的排序来比较。数字的标识符比非数字的标识符优先层级低。若开头的标识符都相同时，栏位比较多的先行版本号优先层级比较高。范例：1.0.0-alpha < 1.0.0-alpha.1 < 1.0.0-alpha.beta < 1.0.0-beta < 1.0.0-beta.2 < 1.0.0-beta.11 < 1.0.0-rc.1 < 1.0.0。

12. 除去API版本号的修订，PPT，文档的版本控制也有相当的必要，参考如上几个规则，制定PPT等相关配置文档版本号规则如下：

自行迭代时的成果一般作为修订号Z出现，经过预评审后根据其评审内容修改的版本号定义为次版本号Y，最后就是正式评审后的正式版本号X

# 3 github配置管理

## 3.1配置界面



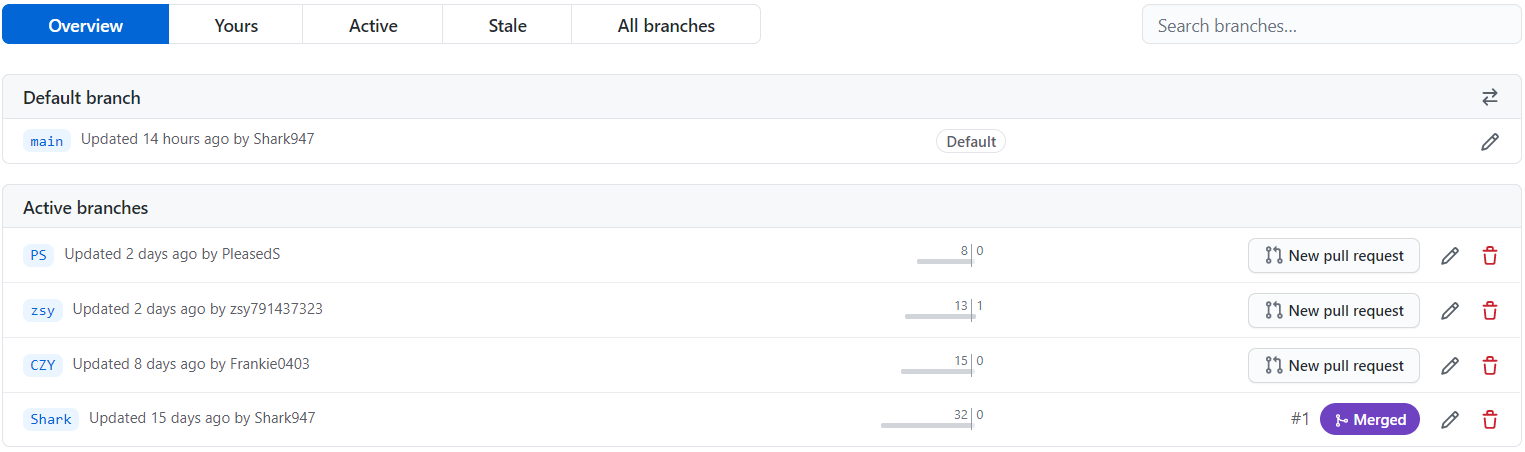
由项目团队，项目名称以及项目当前分支，操作，目录等构成,基本的clone，commit，push就不介绍了

## 3.2管理流程

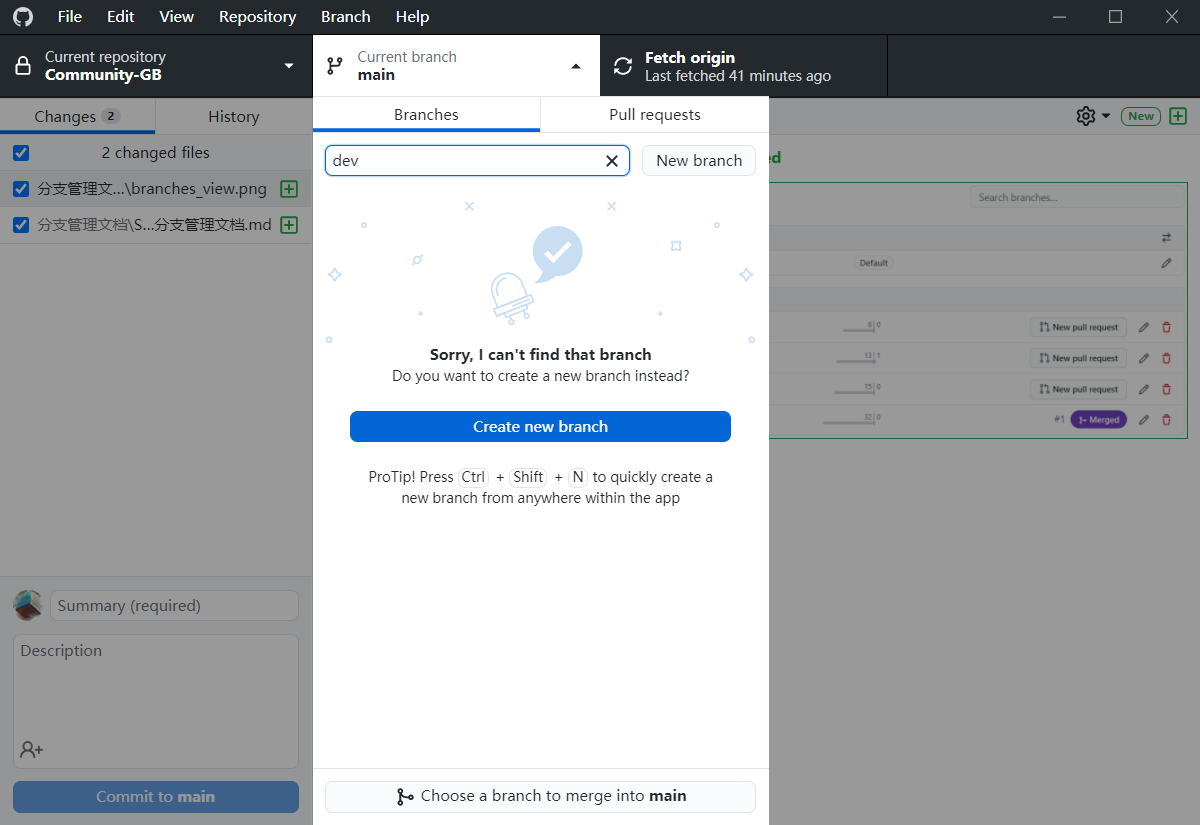
* + 1. 组长分配相应任务
    2. 组员确认需求
    3. 编写文档时先编写版本控制表
    4. 修改完后交由审核人审核，审核通过经由批准人批准上传github
    5. 创建分支
    6. 合并分支

### 3.2.1 创建分支git操作（以git desktop为案例）

这里可以查看总体分支的数量以及状态，甚至于删除对应的分支。

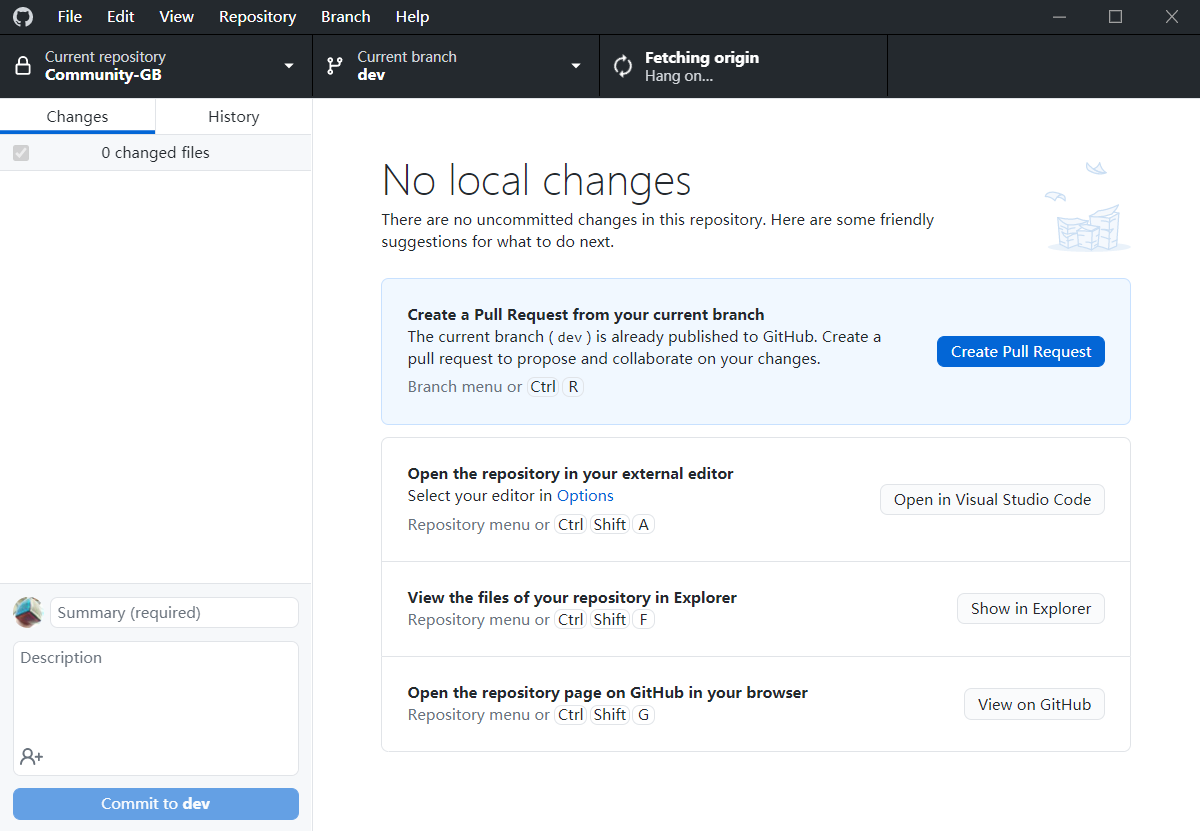


这里可以使用git desktop来建立新的分支



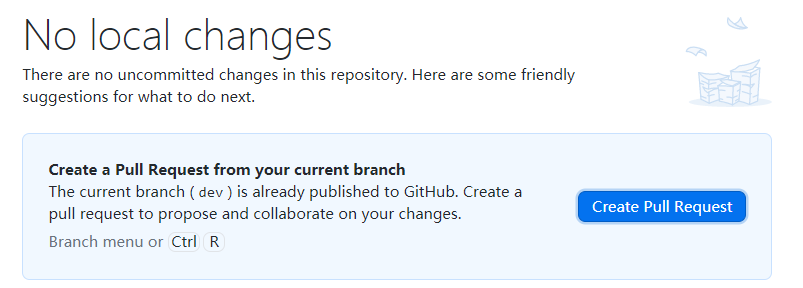
### 3.2.2 管理分支git操作

根据界面按钮先进行commit后push，最后pull request归并分支

****

### 3.2.3 合并分支git操作

成功提交到我们的远程分支仓库后就可以准备合并分支，将提交的内容归并到主分支中了



根据打开的网页（github网页）进行分支归并，也可以查看要归并的分支做了哪些修改

