**软件工程项目研究报告**

**项目名称：学习成果分享平台**

**第二组**

**组长：方琳**

**组员：俞凌飞、卫梦琪、晏思颖**

**报告更新日志**

3.8 确认项目名称、功能需求。

3.15 搭建软件需求规格说明框架，确定本项目所采取的过程模型。

3.18 加入系统用例图，选择两个用例图撰写用例文本。

3.28 加入系统E-R图

4.1 加入用UML活动图建立的业务流程模型和用UML状态图建立的业务元素行为模型

4.19 修正流程图和状态图、加入体系架构

5.04 加入软件质量保证模块，确定编码规范和代码检查工具，修改体系架构图。

5.13 加入用UML类图对项目的建模。

目录

[软件需求规格说明 4](#_Toc40281993)

[1. 引言 4](#_Toc40281994)

[1.1 目的 4](#_Toc40281995)

[1.2 适用范围 4](#_Toc40281996)

[1.3 参考资料 4](#_Toc40281997)

[1.4 术语和缩略语 4](#_Toc40281998)

[2. 系统概述 4](#_Toc40281999)

[2.1 产品描述 4](#_Toc40282000)

[2.2 产品功能 5](#_Toc40282001)

[2.3 系统用例图 5](#_Toc40282002)

[2.4 系统用例文本描述 6](#_Toc40282003)

[2.5 E-R图 7](#_Toc40282004)

[2.6 UML状态图 7](#_Toc40282005)

[2.6.1学习成果发布状态图 7](#_Toc40282006)

[2.6.2 举报评论状态图 8](#_Toc40282007)

[2.7 UML活动图 9](#_Toc40282008)

[2.8 体系架构 9](#_Toc40282009)

[2.8.1 开发平台 9](#_Toc40282010)

[2.8.2 系统架构图 10](#_Toc40282011)

[2.9 UML类图 11](#_Toc40282012)

[2.10面向对象设计思路 11](#_Toc40282013)

[3. 功能性需求分类 12](#_Toc40282014)

[3.1 管理员角色 12](#_Toc40282015)

[3.1.1 管理员登录 12](#_Toc40282016)

[3.1.2 创设科目标签 12](#_Toc40282017)

[3.1.3 筛选动态 12](#_Toc40282018)

[3.1.4 发布通知 12](#_Toc40282019)

[3.1.5 处理举报 12](#_Toc40282020)

[3.2 用户角色 12](#_Toc40282021)

[3.2.1 注册登录 12](#_Toc40282022)

[3.2.2 发表动态、上传文件 12](#_Toc40282023)

[3.2.3 查看动态 12](#_Toc40282024)

[3.2.4 评论动态 13](#_Toc40282025)

[3.2.5 举报功能 13](#_Toc40282026)

[4. 产品非功能性需求 13](#_Toc40282027)

[4.1 外部接口说明 13](#_Toc40282028)

[4.2 性能需求 13](#_Toc40282029)

[4.3 属性 13](#_Toc40282030)

[4.4 系统运行环境 13](#_Toc40282031)

[4.5 编码规范 13](#_Toc40282032)

[4.5.1 编码 13](#_Toc40282033)

[4.5.2 注释 13](#_Toc40282034)

[4.5.3 空格 13](#_Toc40282035)

[4.5.4 空行 14](#_Toc40282036)

[4.5.5 断行 14](#_Toc40282037)

[4.5.6 命名 14](#_Toc40282038)

[4.5.7 import 14](#_Toc40282039)

[4.5.8 赋值 15](#_Toc40282040)

[4.5.9 分支和循环 15](#_Toc40282041)

[4.5.10 文件的使用 15](#_Toc40282042)

[4.6 代码测试工具 15](#_Toc40282043)

[附录 16](#_Toc40282044)

[软件项目计划 17](#_Toc40282045)

[1. 过程模型 17](#_Toc40282046)

# 软件需求规格说明

# 1. 引言

## 1.1 目的

疫情期间，学生开启线上学习模式。为了督促大家每日完成学习任务，促进学习热情，激发学习动力，营造良好的学习氛围，我们计划开发一个可以分享学习成果的互动平台。在此平台上，大家可以分享自己的学习成果，可以看到不同科目下同学们发表的“今日学习成果”的分享动态，还可以线上交流与讨论。

## 1.2 适用范围

适用于信机学院2017级的想要了解他人的学习情况、获得学习动力的大学生。（目前仅限PC端）

## 1.3 参考资料

## 1.4 术语和缩略语

# 2. 系统概述

## 2.1 产品描述

本项目主要研究学习成果分享平台的开发与功能实现，该平台可在PC端使用，使用Visual Studio进行具体的项目开发。

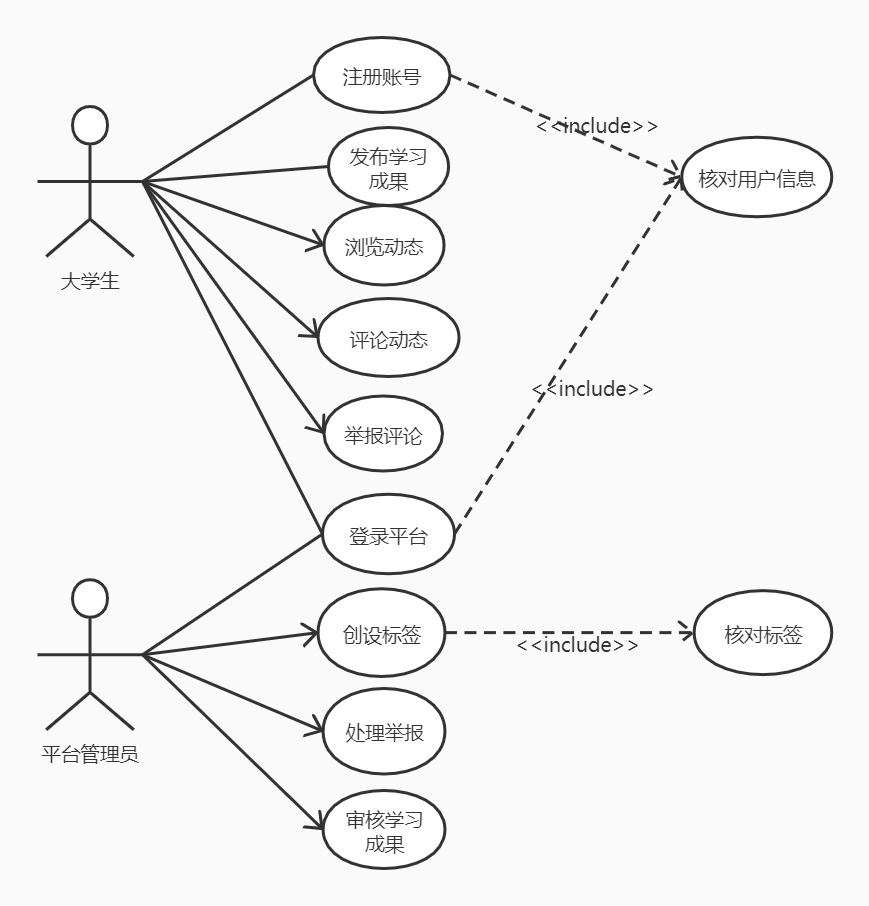
（1）用户设定为信机学院2017级的大学生，可对发布的学习动态进行科目分类、在发表学习动态时还可以上传文件，并对该学习动态所属的科目进行标记；除此之外，用户可以在登录之后选择想要查看的科目，从而与对应科目的动态用户进行信息交互；若是发现不适宜的内容，用户可以向管理员举报。

（2）管理员是负责平台管理的专门人员，其主要工作有：根据实际需要实时创设科目标签、对用户发布的动态进行筛选、发布通知以及处理举报。

## 2.2 产品功能

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 功能名称 | 对应角色 | 功能简要描述 |
| 1 | 管理员登录 | 管理员 | 管理员通过已经存在的工号及密码进行登录，进入管理员主页界面。 |
| 2 | 创设科目标签 | 管理员可实时更新增设科目标签。 |
| 3 | 筛选动态 | 管理员可以对用户发布的动态进行筛选，动态是否能成功发布由管理员审查的结果决定。 |
| 4 | 发布通知 | 管理员可以通过发布通知来告知用户系统更新、举报是否已处理等消息。 |
| 5 | 处理举报 | 管理员对用户发出的举报会进行相应的处理。 |
| 6 | 注册登录 | 用户 | 用户需要先进行注册之后，才能登录进入主页面。 |
| 7 | 发表动态  上传文件 | 已注册的用户可以根据自己的需要发表动态、上传文件。 |
| 8 | 查看动态 | 登录后，用户可以根据想要了解的科目，查看别人发表的学习动态。 |
| 9 | 评论动态 | 登录后，用户可以在其他用户发表的学习动态下进行文字的评论。 |
| 10 | 举报功能 | 对于不适宜的内容，用户可以向管理员举报。 |

## 2.3 系统用例图



## 2.4 系统用例文本描述

**2.4.1 用例一**

用例名称：发布动态

参与者：用户、管理员

前置条件：用户已经登录、管理员已经登录平台

基本流描述：

1. 用户选择“发布动态”时进入该流程；  
2. 用户编辑动态内容，如有需要上传文件；  
3. 用户勾选与动态内容相匹配的学科标签；  
4. 用户点击“确认发表”；  
5. 管理员收到用户发表的动态并审核其内容；  
6. 管理员确认发表。

扩展流描述：

1. 基本流第5步中若管理员审核动态内容不合适，则不予发表。

后置条件：确认发表的动态被系统标记并存档

**2.4.2 用例二**

用例名称：浏览动态

参与者：用户

前置条件：用户已经登录平台

基本流描述：

1. 当用户选择“浏览动态”时，进入该用例流程

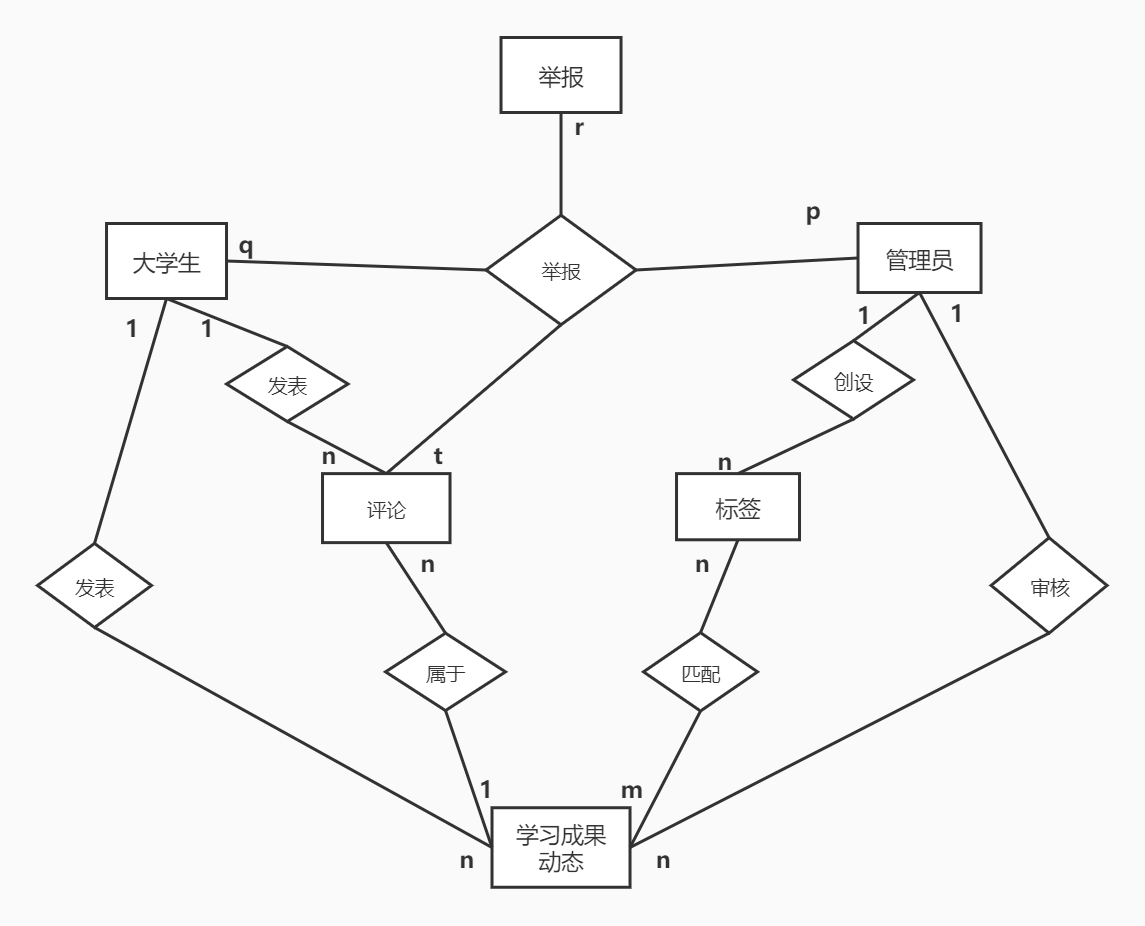
2. 用户选择一种学科分类标签，系统提供该学科的已发表动态

3. 用户浏览动态，根据喜好选择性下载文件

扩展流描述：

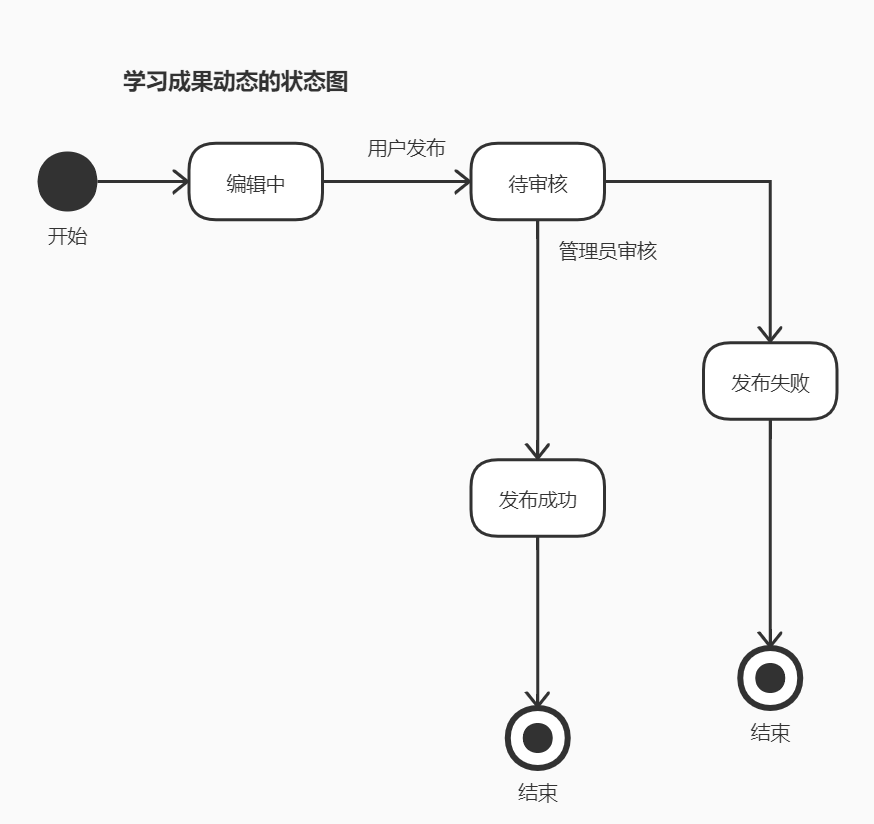
1. 基本流第3步中，用户在浏览动态时，可以在动态下方进行动态评论

## 2.5 E-R图

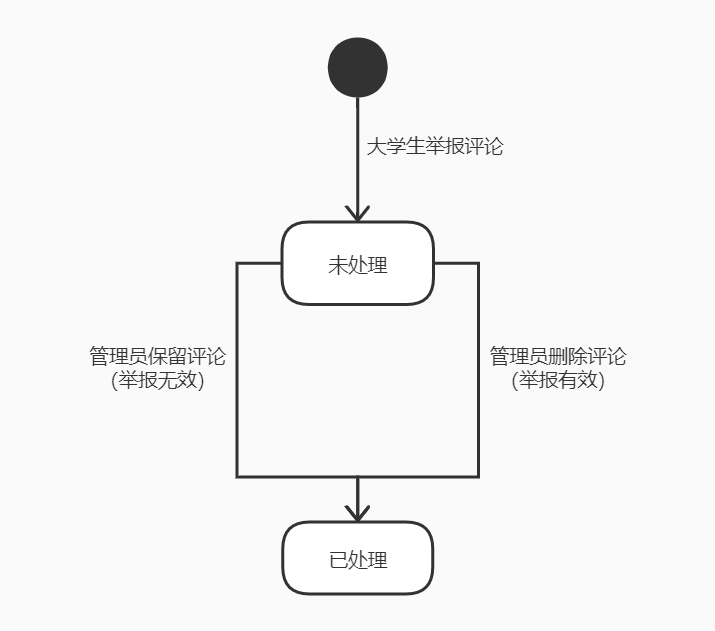


## 2.6 UML状态图

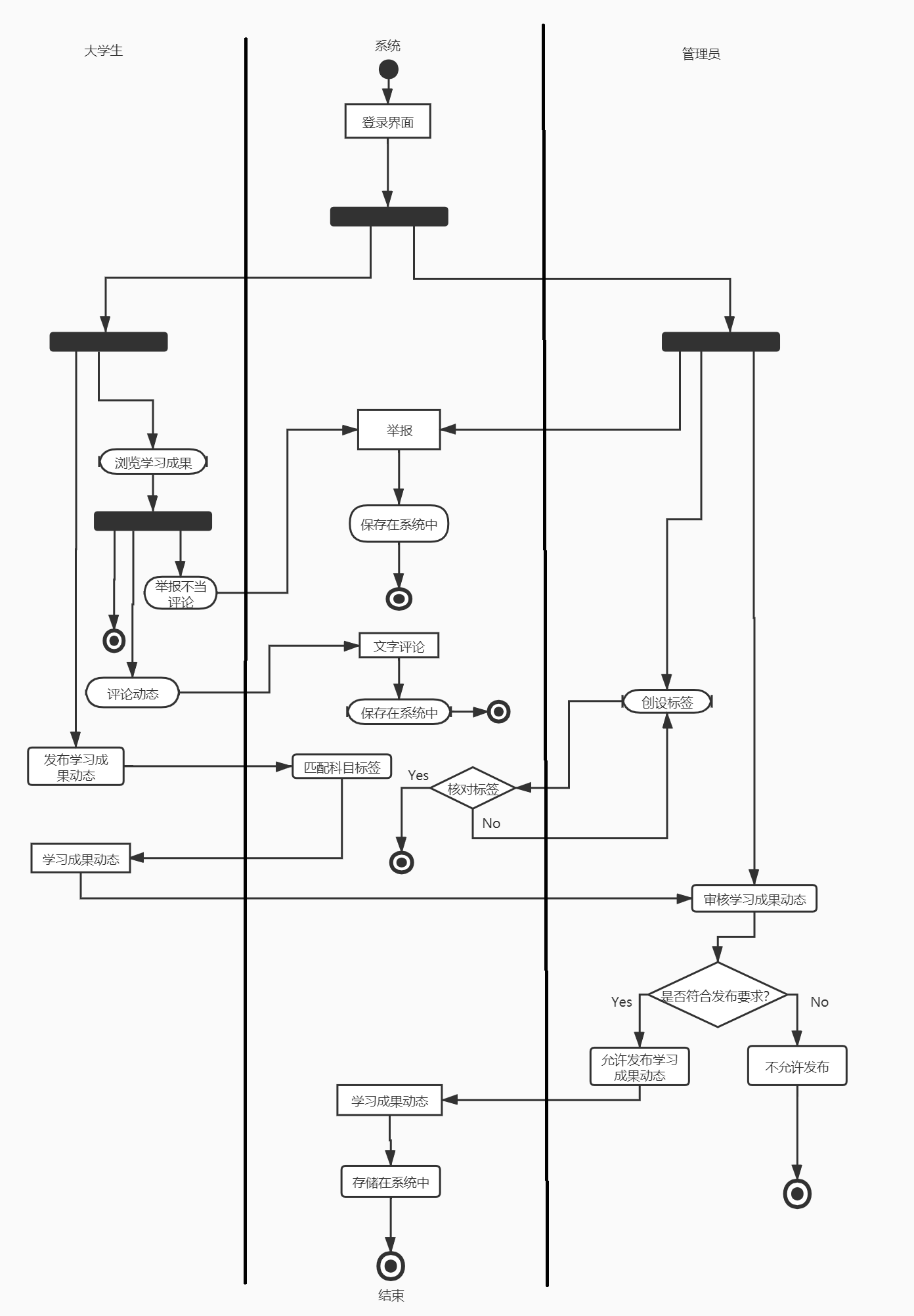
### 2.6.1学习成果发布状态图

****

### 2.6.2 举报评论状态图

****

## 2.7 UML活动图

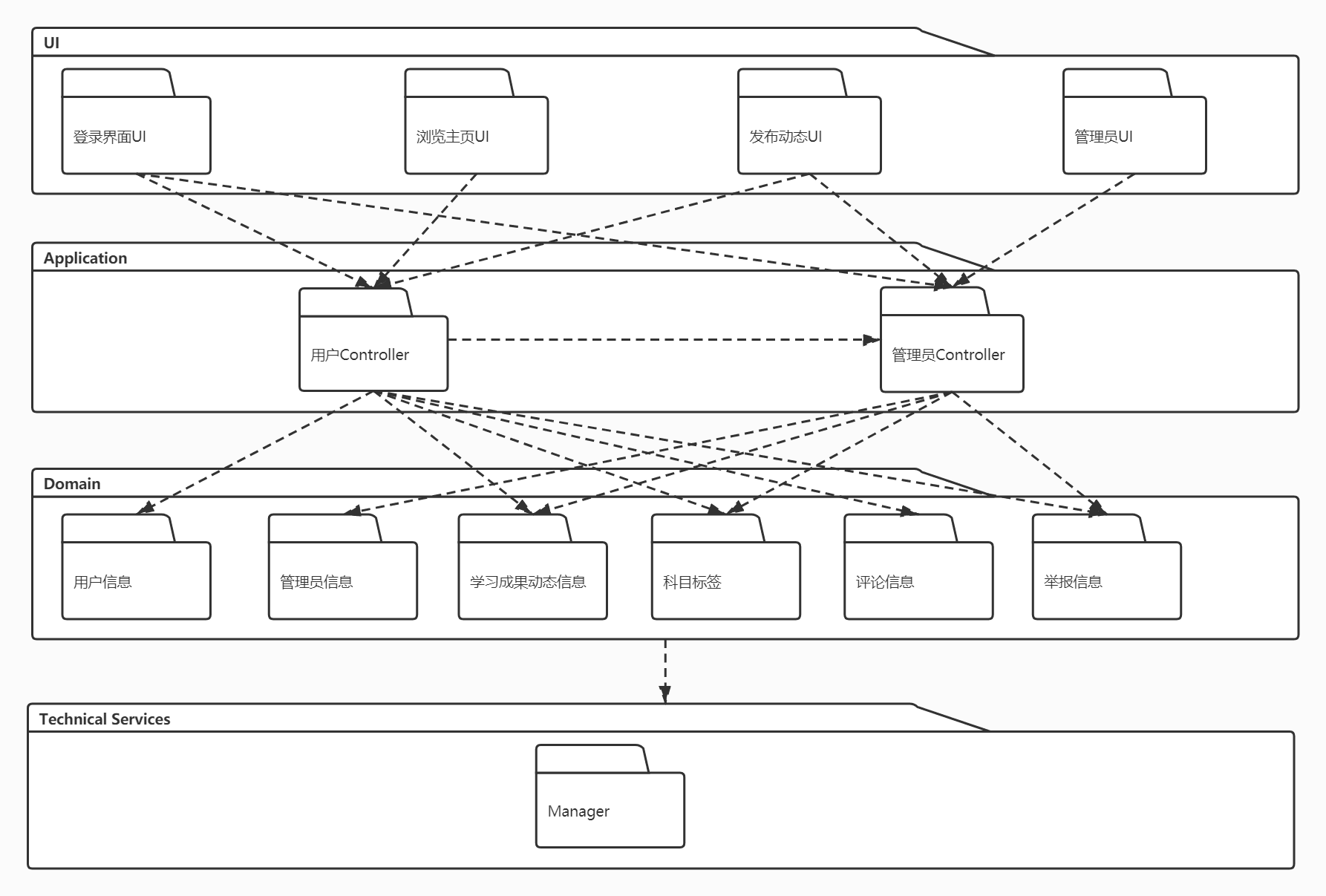


## 2.8 体系架构

### 2.8.1 开发平台

选用Python作为开发语言，Python IDLE作为开发工具，Django作为框架。使用GitHub进行配置管理和协作开发。

### 2.8.2 系统架构图



程序系统架构分为四个层次，分别是UI层、Application层、Domain层和Technical Services层。

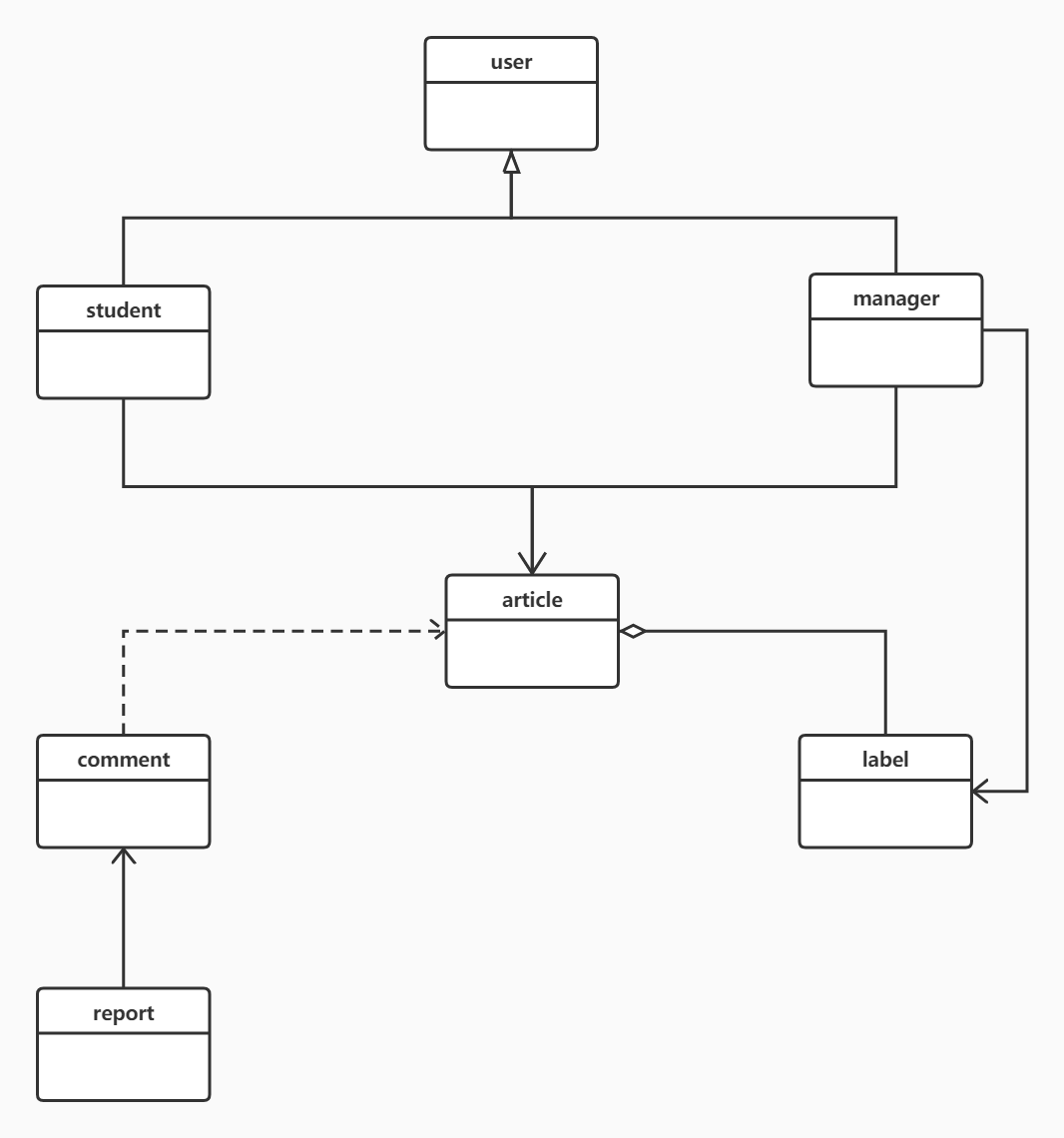
程序主体UI分为四部分，分别是管理员UI、登录界面UI、浏览主页UI和发布动态UI。登录界面UI、浏览主页UI和发布动态UI是用户进行活动的主要实现界面，管理员审核动态、处理举报等界面归入管理员UI。

Application层分为用户controller和管理员controller，两者各自调用不同数据进行功能实现。

Domain层呈现的是需要存储的数据，包括用户信息、管理员信息、学习成果动态信息、科目标签、评论信息、举报信息等。

Technical Services层主要是Django自带的访问数据库的接口：manager(管理器)。

## 2.9 UML类图



## 2.10面向对象设计思路

根据本项目的实际需求，我们总结划分出以下类：

1. user类：是用户父类，由普通用户（大学生）类和管理员类来继承该父类。
2. article类：学习成果动态类，与label（标签类）是聚合关系，学习成果的组成部分中包含标签。与管理员和大学生是关联关系，学习成果动态会被大学生和管理员创建实体对象，修改具体内容，故为关联关系。
3. comment类：评论类。与学习成果动态是依赖的关系。学习成果动态可以没有评论。
4. report类：举报类。举报类与评论类是关联关系。report类中会出现comment类的成员变量，以改变comment的某些状态，因此为关联关系。
5. label类：标签类。管理员会创建标签，改变不同标签对象的某些属性，故为关联关系。

# 3. 功能性需求分类

## 3.1 管理员角色

### 3.1.1 管理员登录

管理员通过已经存在的工号及密码进行登录，进入管理员主页界面。

### 3.1.2 创设科目标签

管理员登录后可创设用户需要的所有科目标签，例如：软件工程、计算机教学法、硬件类课程、Andriod技术开发等，以供用户在发表动态时勾选标记动态的科目类型。

### 3.1.3 筛选动态

管理员登录后，可接收到由用户发表的动态申请，根据用户发表的动态内容，筛选与学习相关的动态与文件并同意发表，如发现用户发表与学习无关的动态或文件则不予通过。

### 3.1.4 发布通知

管理员登录后可以发表通知动态，可针对全体和个人（选择群发或@某个用户）。具体内容可以是平台使用指南、发布动态流程指导、动态问题处理反馈等。

### 3.1.5 处理举报

用户使用评论回复功能时，发现有不文明用语（包括：低俗用语、恶意引战、淫秽色情、人身攻击、其它不文明用语等），可使用举报功能，而管理员登录后可以处理此类举报，对于被举报用户言论进行删除，净化平台的信息交互环境。

## 3.2 用户角色

### 3.2.1 注册登录

未注册学生可使用姓名、学号进行注册。注册成功后，使用学号及密码进行登录，进入个人主页界面。

### 3.2.2 发表动态、上传文件

已注册学生登录后可在个人主页界面上选择科目标签发表学习动态，上传学习文件，例如：个人学习笔记、学习计划及成果等。用户动态及文件在发表时默认为所有用户可见。

### 3.2.3 查看动态

所有用户可通过选择科目标签对所有动态进行筛选，从而查看所选标签下的相关动态及文件。

### 3.2.4 评论动态

所有用户可对感兴趣的动态进行文字评论，也可以在别人的评论下进行文字回复，从而进行信息交互，提高平台的活跃度。

### 3.2.5 举报功能

所有用户使用评论回复功能时，发现有不文明用语（包括：低俗用语、恶意引战、淫秽色情、人身攻击、其它不文明用语等），可以使用举报功能，向系统管理员进行举报，以维护平台和谐稳定的使用。

# 4. 产品非功能性需求

## 4.1 外部接口说明

## 4.2 性能需求

## 4.3 属性

## 4.4 系统运行环境

## 4.5 编码规范

### 4.5.1 编码

所有的 Python 脚本文件都应在文件头标上 # -\*- coding:utf-8 -\*- 。设置编辑器，默认保存为 utf-8 格式。

### 4.5.2 注释

1.由 # 开头的“真正的”注释，说明选择当前实现的原因以及这种实现的原理和难点；  
 2.文档字符串”””……””” ：表明如何使用这个包、模块、类、函数（方法），甚至包括使用示例和单元测试；  
 3.坚持适当注释原则。写注释要注重质量而不是数量。

### 4.5.3 空格

在行首的空格称为前导空格。非前导空格在 Python 代码中没有意义，但适当地加入非前导空格可以增进代码的可读性，加空格可以参考如下几个准则：

1.在二元算术、逻辑运算符前后加空格：如 a = b + c；

2.在一元前缀运算符后不加空格，如 if !flg: pass；

3.“:”用在行尾时前后皆不加空格，如分枝、循环、函数和类定义语言；用在非行尾时两端加空格，如 dict 对象的定义 d = {‘key’ : ’value’}；

4.括号前后不加空格；

5.逗号后面加一个空格，前面不加空格。

### 4.5.4 空行

适当的空行有利于增加代码的可读性，加空行可以参考如下几个准则：  
1.在类、函数的定义间加空行；  
2.在 import 不同种类的模块间加空行；  
3.在函数中的逻辑段落间加空行，即把相关的代码紧凑写在一起，作为一个

逻辑段落，段落间以空行分隔。

### 4.5.5 断行

规范仍然坚持行的最大长度不得超过78个字符的标准。折叠长行的方法有以下几种方法：  
 1.为长变量名换一个短名；  
 2.在括号（包括圆括号、方括号和花括号）内换行。如果行长到连第一个括号内的参数都放不下，则每个元素都单独占一行；  
 3.在长行加入续行符强行断行，断行的位置应在操作符前，且换行后多一个缩进。

### 4.5.6 命名

1.常量：常量名所有字母大写，由下划线连接各个单词；  
2.变量：变量名全部小写，由下划线连接各个单词。私有类成员使用单一下划线前缀标识，多定义公开成员，少定义私有成员。变量名不应带有类型信息；  
3.函数：函数名的命名规则与变量名相同；  
4.类：类名单词首字母大写，不使用下划线连接单词；  
5.模块：模块名全部小写，对于包内使用的模块，可以加一个下划线前缀；  
6.包：包的命名规范与模块相同；  
7.缩写：命名应当尽量使用全拼写的单词，缩写的情况有如下两种：  
 1） 常用的缩写，在命名时只大写首字母；  
 2） 命名中含有长单词，对某个单词进行缩写，使用约定成俗的缩写方式。   
8.特定命名方式：指 \_\_xxx\_\_ 形式的系统保留字命名法。这种形式的变量是只读的。

### 4.5.7 import

import 语句有以下几个原则需要遵守：

1.import 的次序，先 import Python 内置模块，再 import 第三方模块，最后 import 自己开发的项目中的其它模块；这几种模块中用空行分隔开来；

2.一条 import 语句 import 一个模块;

3.当从模块中 import 多个对象且超过一行时，使用断行法;

4.不要使用 from module import \*，除非是 import 常量定义模块，或其它你确保不会出现命名空间冲突的模块。

### 4.5.8 赋值

1.对于赋值语言，注意不要做无谓的对齐;

2.=前后要有空格.

### 4.5.9 分支和循环

对于分枝和循环，有如下几点需要注意的：

1.不要写成一行；

2.条件表达式的编写应该足够简洁明了，具备专业性;

如以下形式的条件表达式是拙劣的：

if var != None: do\_something()

if var != False: do\_something()

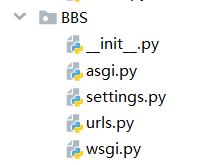
上面的语句应该写成：

if seq: do\_somethin()

if var: do\_something()

3)多使用循环语句的 else 分句，以简化代码。

### 4.5.10 文件的使用



1.BBS/\_\_init\_\_.py：一个空文件，告诉Python这个目录被认为是一个Python包。

2.BBS/asgi.py: 作为项目运行在ASGI兼容的web服务器上的入口。

3. BBS/settings.py: Django项目的配置文件。

4. BBS/urls.py：Django项目的URL声明，就像网站的“目录”。

5. BBS/wsgi.py：作为项目运行在WSGI兼容的web服务器上的入口。

## 4.6 代码测试工具

1.代码检查工具：profile（测试代码的规范性）;

2.Django自带的测试工具：client（测试代码的正确性）。

# 附录

# 软件项目计划

## 1. 过程模型

本项目拟使用迭代式开发方式中的增量模型进行开发。原因如下：

（1）本项目的核心是学习成果分享平台的开发与功能实现，在这样一个平台中，功能

较多，且各个功能之间的优先权不同。比如发表动态和上传文件就是最核心的功能，评论功能及举报功能次之。利用增量模型，具有最高优先权的核心增量构件将会被最先交付，而随着后续构件不断被集成进系统，这个核心构件将会受到最多次数的测试。这意味着软件系统最重要的心脏部分将具有最高的可靠性，这将使得整个软件系统更具健壮性。

（2）采用增量模型，可使得开发初期用户只需对软件需求进行大概的描述，对于需求

的细节性描述则可以延迟到增量构件开发时进行，以增量构件为单位逐个地进行需求补充。这使得我们可以尽快地着手项目的开发。但是增量模型的缺点是灵活性较大，容易退化为边做边改的模型，我们会尽量克服这一切困难。

（3）不采用瀑布模型或快速原型化模型是因为这种模型的目标是一次性把一个满足所

有需求的产品提交给用户。而作为小组作业，以学习为目的的共同开发一个项目的过程中，制定阶段性的目标有利于及时检测学习成果，也可鼓舞团队士气以及降低项目的风险。