# 1、系统概述

**1.1、系统简介**

简要概述系统的基本情况和背景，还包括概要设计中术语的介绍等。

本系统是一个小巧轻量、集发布和讨论于一体的论坛平台；基于北邮人论坛的使用和周围同学的反馈背景，让用户体验更加简约流畅。

**1.2、系统目标**

概述概要设计要实现的目标，包括功能目标、性能目标等。

功能目标：

1、论坛用户可以注册和登录，并能够发布文章和动态、查看用户消息

2、论坛用户可以对论坛上的文章和动态进行评论、点赞和收藏

3、管理端可以对用户发送系统消息和禁用账户

4、管理端可以对论坛上的帖子和评论进行审核、排序和删除

性能目标：

1、精度：用户上传的文字和图片信息能够被系统正确识别、处理和输出

2、时间特性：用户能够快速访问和使用系统，系统能在短时间内快速响应用户的请求

3、灵活性：系统能够适应不同用户和不同场景下的使用需求、技术和环境的变化

4、故障处理：对于网络、数据库故障和安全漏洞能够及时发现和维护

**1.3、系统运行环境**

包括对硬件平台、操作系统、数据库系统、编程平台、网络协议等的描述。

硬件平台：Legion Y7000P 2019、Dell G15 5511、MacBook Air(M1)

操作系统：Windows、Mac OS、Linux

前端框架：Vue.js + Vue Route + Naïve UI + Element Plus + Less + Axios

后端框架：FastAPI+SQLAlchemy

数据库：Sqlite

编程平台：VSCode

网络协议：TCP/IP

**1.4、开发环境**

列举进行系统分析、程序设计和程序开发时要使用的工程工具和开发语言。应描述每一工具软件的名称、版本等。

工程工具：

集成开发环境：VSCode、PyCharm

版本控制工具：Git

自动化构建工具：Vite

调试工具：Volar、Unicorn、Chrome浏览器

截止到项目发布时的最新版

开发语言：

Vue：3.2.47

Python：3.9.12

# 2、总体结构设计

**2.1、软件结构**

按照不同功能进行整体结构层次的划分，并使各层功能相对独立。以图形方式给出软件系统的子系统（或软件包）划分，模块划分，子系统间、模块间关系等，并用接口来描述各模块之间的调用关系，给出各模块之间的松散耦合关系。

**2.2、设计思想**

结合上图阐述软件的基本设计思想和理念。

# 3、模块设计

在此以列表形式说明各模块的名称、功能、接口等信息。

**3.1、模块1**

详细描述各功能模块的功能、接口及内部实现。

3.1.1、功能描述

简明有条理地叙述该模块功能，也可画功能流程图。

3.1.2、接口描述

描述与其他功能模块的每个接口的名称、输入信息、输出信息、异常处理等。

3.1.3、数据结构描述

本功能区域使用的数据结构。

3.1.4、实现思路

给出实现该模块的基本思路，包括对模块内部结构、算法、编程方法等的初步设想。

3.1.5、其他

选写另外一些有必要叙述的部分，如模块的表现形式（以何种形式运行，如服务程序、应用程序、数据库脚本等等）、有无人机交互界面（有则给出其类型和设计原则）等。

**3.2、模块2**

……

# 4、数据库与数据结构设计

**4.1、 数据库及数据表**

描述所使用的数据库系统,及数据库和数据表设计。如果系统不以数据库方式存储数据则可省略。

数据库系统：Sqlite

数据库和数据表设计：

用户表（users）:

id：用户ID，主键

username：用户名，唯一

email：电子邮件，唯一

password\_hash：密码哈希值

avatar\_url：头像URL

introduction：个人介绍

create\_time：注册时间

update\_time：最后一次更新时间

is\_admin：是否为管理员，布尔类型

is\_active：是否已被激活，布尔类型

帖子表（posts）

id：帖子ID，主键

title：帖子标题

content：帖子内容

create\_time：帖子创建时间

update\_time：最后一次更新时间

is\_deleted：是否已被删除，布尔类型

user\_id：发帖用户ID，外键

评论表（comments）

id：评论ID，主键

content：评论内容

create\_time：评论创建时间

update\_time：最后一次更新时间

is\_deleted：是否已被删除，布尔类型

user\_id：评论用户ID，外键

post\_id：所属帖子ID，外键

parent\_id：父评论ID，外键，可以为空

关注表（follows）

id：关注ID，主键

follower\_id：关注者ID，外键

followed\_id：被关注者ID，外键

点赞表（likes）

id：点赞ID，主键

user\_id：点赞用户ID，外键

post\_id：被点赞帖子ID，外键

comment\_id：被点赞评论ID，外键，可以为空

通知表（notifications）

id：通知ID，主键

content：通知内容

create\_time：通知创建时间

is\_read：是否已读，布尔类型

user\_id：接收通知的用户ID，外键

**4.2、 数据结构设计**

给出本系统内所使用的每个数据结构的名称、标识符以及它们之中每个数据项、记录、文卷和系的标识、定义、长度及它们之间的层次的或表格的相互关系。

本系统中使用的数据结构包括以下表格：

用户表（users）：

标识符：users

数据项：

id: 用户ID，整数类型，主键，自增长

username: 用户名，字符串类型，唯一，长度为20

email: 电子邮件，字符串类型，唯一，长度为64

password\_hash: 密码哈希值，字符串类型

avatar\_url: 头像URL，字符串类型

introduction: 个人介绍，字符串类型

create\_time: 注册时间，日期时间类型

update\_time: 最后一次更新时间，日期时间类型

is\_admin: 是否为管理员，布尔类型

is\_active: 是否已被激活，布尔类型

帖子表（posts）：

标识符：posts

数据项：

id: 帖子ID，整数类型，主键，自增长

title: 帖子标题，字符串类型

content: 帖子内容，字符串类型

create\_time: 帖子创建时间，日期时间类型

update\_time: 最后一次更新时间，日期时间类型

is\_deleted: 是否已被删除，布尔类型

user\_id: 发帖用户ID，整数类型，外键，关联用户表的id字段

评论表（comments）：

标识符：comments

数据项：

id: 评论ID，整数类型，主键，自增长

content: 评论内容，字符串类型

create\_time: 评论创建时间，日期时间类型

update\_time: 最后一次更新时间，日期时间类型

is\_deleted: 是否已被删除，布尔类型

user\_id: 评论用户ID，整数类型，外键，关联用户表的id字段

post\_id: 所属帖子ID，整数类型，外键，关联帖子表的id字段

parent\_id: 父评论ID，整数类型，外键，可以为空，关联评论表的id字段

关注表（follows）：

标识符：follows

数据项：

id: 关注ID，整数类型，主键，自增长

follower\_id: 关注者ID，整数类型，外键，关联用户表的id字段

followed\_id: 被关注者ID，整数类型，外键，关联用户表的id字段

点赞表（likes）：

标识符：likes

数据项：

id: 点赞ID，整数类型，主键，自增长

user\_id: 点赞用户ID，整数类型，外键，关联用户表的id字段

post\_id: 被点赞帖子ID，整数类型，外键，关联帖子表的id字段

comment\_id: 被点赞评论ID，整数类型，外键，可以为空，关联评论表的id字段

通知表（notifications）：

标识符：notifications

数据项：

id: 通知ID，整数类型，主键，自增长

content: 通知内容，字符串类型

create\_time: 通知创建时间，日期时间类型

is\_read: 是否已读，布尔类型

user\_id: 接收通知的用户ID，整数类型，外键，关联用户表的id字段

这些表之间的关系如下：

用户表（users）和帖子表（posts）是一对多的关系，即一个用户可以创建多个帖子，但一个帖子只能由一个用户创建。

帖子表（posts）和评论表（comments）也是一对多的关系，即一个帖子可以有多个评论，但一个评论只能属于一个帖子。

评论表（comments）和自身之间存在一种多级关系，即一个评论可以有多个子评论（即回复），也可以没有子评论。

用户表（users）和关注表（follows）之间是多对多的关系，即一个用户可以关注多个其他用户，也可以被多个用户关注。

用户表（users）和点赞表（likes）之间也是多对多的关系，即一个用户可以给多个帖子或评论点赞，一个帖子或评论也可以被多个用户点赞。

用户表（users）和通知表（notifications）是一对多的关系，即一个用户可以收到多条通知，但一条通知只能发送给一个用户。

**4.3、 数据存储设计**

给出本系统内所使用的每个数据结构中的每个数据项的存储要求,访问方法、存取单位、存取的物理关系(索引、设备、存储区域)、设计考虑和保密条件。

访问方法：

以上数据结构中的数据项可以通过SQLAlchemy语句进行访问。

存取单位：

以上数据结构中的数据项可以按照单个记录（一行）或整个表格（多行）进行存取。

存取的物理关系：

以上数据结构中的数据项存储在SQLite数据库中，可以使用索引访问。

# 5、接口设计

**4.1、外部接口**

描述需要访问的外部接口的类型、控制方式。

**4.2、内部接口**

描述与其他模块及子系统的接口。

# 6、其他设计

对以上概要设计中未包含项目的说明，如并行处理、安全设计、内存管理等。

安全设计：OAuth2 实现密码哈希与 Bearer JWT 令牌验证