# 1、系统概述

**1.1、系统简介**

BuptHub是一个小巧轻量、集发布和讨论于一体的论坛，旨在加强学生交流互助、促进知识共享和学术讨论。本论坛具有以下功能：

1、论坛用户可以注册和登录，并能够发布帖子、查看系统消息

2、论坛用户可以对论坛上的帖子进行评论、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

平台的建立背景是满足学生社区的交流需求，在传统社交媒体之外开辟出具有特定环境的学生平台，使同学们拥有可以讨论课程内容、分享学习资源、解决问题的专属空间，为同学们提供一个安全、友好的学术交流环境；同时，基于北邮人论坛的使用体验，BuptHub将让用户体验更加简约流畅。

**1.2、术语表**

|  |  |  |
| --- | --- | --- |
| **序号** | **术语或缩略语** | **说明性定义** |
| 1 | Vue.js | 一款用于构建用户界面的 JavaScript 框架 |
| 2 | Vue Route | Vue.js 的官方路由 |
| 3 | Naïve UI , Element Plus | 基于 Vue 3，面向设计师和开发者的组件库 |
| 4 | Less | Leaner Style Sheets是一门向后兼容的 CSS 扩展语言 |
| 5 | Axios | 一个基于 promise 的网络请求库 |
| 6 | FastAPI | 一个使用 Python 3.6+ 用于构建 API 的 web 框架 |
| 7 | SQLAlchemy | 一个python语言实现的针对关系型数据库的ORM库 |
| 8 | ORM | 对象关系映射（Object Relational Mapping，简称ORM）是解决面向对象与关系数据库互不匹配的技术模式 |

**1.3、系统运行环境**

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 Router + 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**

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、数据结构说明

2.1、常量

SECRET\_KEY：

常量名称：SECRET\_KEY

所在目录：backend/app/core/security.py

常量取值：生成的32位随机密钥

功能说明：用于生成安全令牌（token）的密钥。

ALGORITHM：

常量名称：ALGORITHM

所在目录：backend/app/core/security.py

常量取值：HS256

功能说明：指定生成 JWT token 使用的算法，使用 HMAC-SHA256 算法进行签名。

ACCESS\_TOKEN\_EXPIRE\_MINUTES：

常量名称：ACCESS\_TOKEN\_EXPIRE\_MINUTES

所在目录：backend/app/core/security.py

常量取值：60 \* 24 \* 7

功能说明：指定 JWT token 的过期时间，单位为分钟。在本程序中设置为7天有效期。

UPLOAD\_FOLDER：

常量名称：UPLOAD\_FOLDER

所在目录：backend/app/routers/users.py

常量取值：文件路径 "../frontend/demo1/public/static/avatar/"

功能说明：指定用户头像文件的上传路径，即将用户上传的头像文件存储在该路径下。

SQLALCHEMY\_DATABASE\_URL：

常量名称：SQLALCHEMY\_DATABASE\_URL

所在目录：backend/app/sql\_app/database.py

常量取值："sqlite:///./sql\_app.db"

功能说明：指定 SQLite 数据库文件的路径，以及数据库的名称。

emailh：

常量名称：emailh

所在目录：backend/app/mail.py

常量取值：smtp.qq.com

功能说明：指定邮箱服务器地址，即用于发送电子邮件的服务器的地址。

emailu：

常量名称：emailu

所在目录：backend/app/mail.py

常量取值：邮箱用户名

功能说明：指定邮箱用户名，即用于发送电子邮件的邮箱的用户名。

emailpsw：

常量名称：emailpsw

所在目录：backend/app/mail.py

常量取值：邮箱密码

功能说明：指定邮箱密码，即用于发送电子邮件的邮箱的密码。

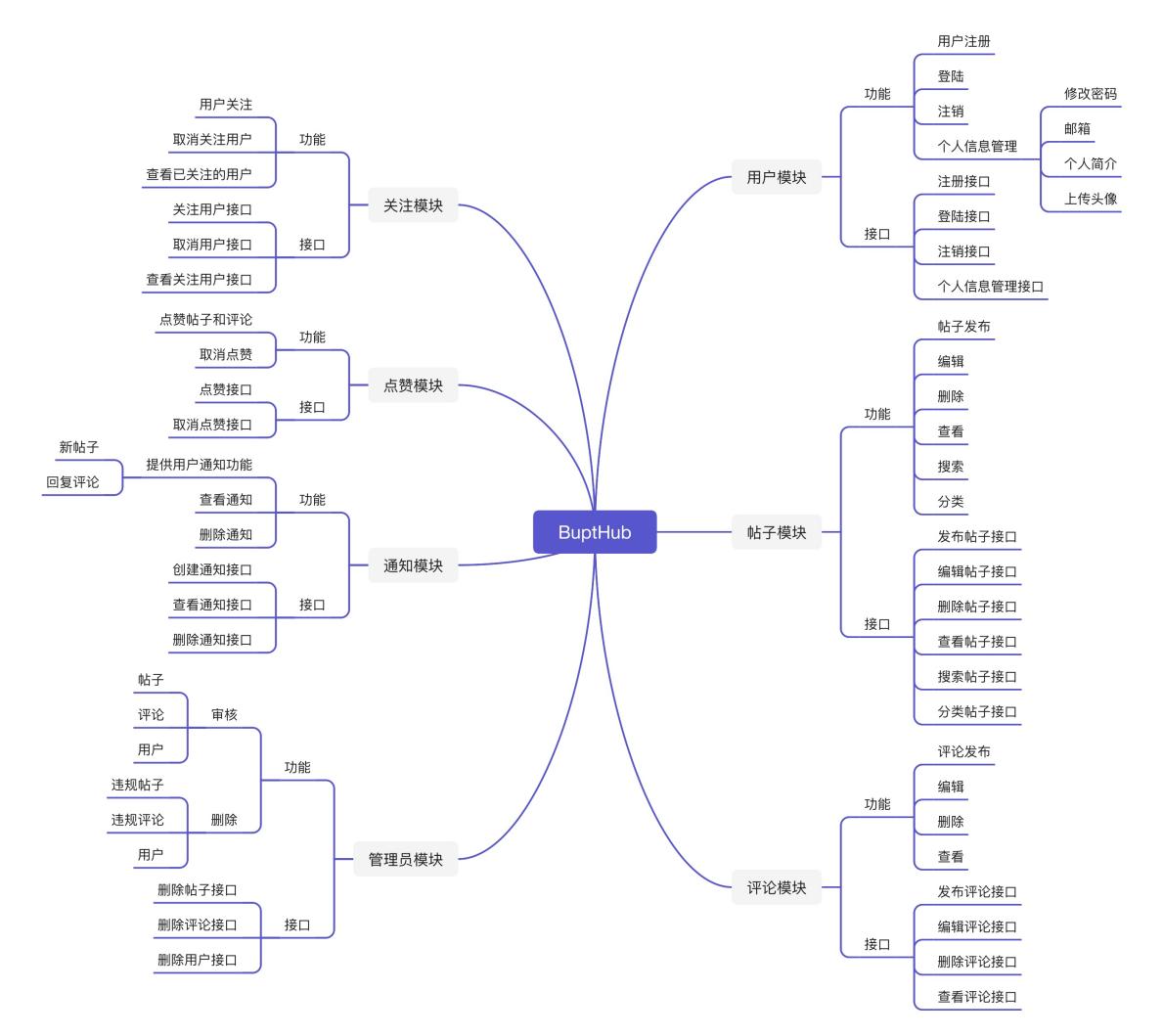
2.2、变量

2.3、数据结构

变量和数据结构的详细信息请参考5.数据库设计章节

# 3、模块设计

**3.1、软件结构**

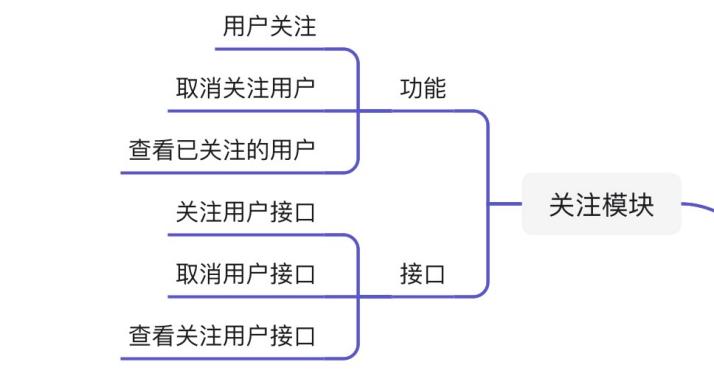
****

**3.2、功能设计说明**

结合上图阐述软件的基本设计思想和理念。

**3.3、模块1**

3.3.1、设计图



3.3.2、功能描述

关注其他用户、取消关注用户、查看已关注的用户

3.3.3、输入数据

关注和取消关注用户输入数据是目标用户的用户名或用户ID；查看已关注用户无额外输入数据，用户直接点击相关按钮或链接。

有效性检验规则有：

存在性检验：验证目标用户是否存在于用户关注列表中。

合法性检验：验证用户输入的用户名或用户ID是否符合论坛的命名规则和格式要求。

从数据库中的用户表（users）和关注表（follows）获取数据，获取条件是用户注册成功后将其信息存储在用户表；以及用户成功关注或取消关注其他用户时，相应的关注关系将被添加或删除到关注表。

3.3.4、输出数据

产生的数据：记录了用户关注其他用户的关系，包括关注者用户ID和被关注者用户ID的关注表（follows）；以及包含了所有注册用户的个人信息和用户ID的用户表（users）。

数据的表现形式：关注表（follows）记录了每个用户关注其他用户的关系，通过关注者用户ID和被关注者用户ID之间的对应关系；用户表（users）包含了每个注册用户的个人信息和用户ID。

3.3.5、数据设计

1. 数据结构名称：User

- 功能说明：表示用户的个人信息和关注关系。

- 具体数据结构说明：

- 定义：用户的个人信息和关注关系，包括用户ID、用户名、年龄、性别、学校、专业、关注列表等。

- 注释设计：该数据结构可以使用类或结构体来表示，在编程语言中定义对应的属性和方法。关注列表可以使用数组、链表或集合等数据结构来存储关注的用户ID。

2. 数据库表：UserTable

- 数据存储设计：

- 数据文件名：user\_table.db

- 数据存储格式：关系型数据库表格形式

- 数据项及属性：

- 用户ID：唯一标识用户的ID，例如字符串类型。

- 用户名：用户的用户名，例如字符串类型。

- 年龄：用户的年龄，例如整数类型。

- 性别：用户的性别，例如字符串类型。

- 学校：用户所在学校，例如字符串类型。

- 专业：用户的专业，例如字符串类型。

- 关注列表：存储用户关注的其他用户的ID列表，可以是数组或字符串类型。

3. 数据库表：FollowingTable

- 数据存储设计：

- 数据文件名：following\_table.db

- 数据存储格式：关系型数据库表格形式

- 数据项及属性：

- 关注者用户ID：关注其他用户的用户ID，例如字符串类型。

- 被关注者用户ID：被关注的用户ID，例如字符串类型。

3.3.6、算法和流程

1. 关注其他用户：

1. 验证输入数据的有效性，包括检查目标用户是否存在和用户输入的格式是否符合要求。

2. 如果输入数据无效，返回错误提示信息。

3. 检查当前用户是否已经关注了目标用户，以避免重复关注。

4. 将关注关系添加到关注表中，将当前用户的ID和目标用户的ID存储为一条记录。

5. 返回成功的提示信息，表示关注操作已成功。

B. 取消关注用户：

1. 验证输入数据的有效性，包括检查目标用户是否存在和用户输入的格式是否符合要求。

2. 如果输入数据无效，返回错误提示信息。

3. 检查当前用户是否已经关注了目标用户。

4. 如果关注关系存在，将该关注关系从关注表中删除。

5. 返回成功的提示信息，表示取消关注操作已成功。

C. 查看已关注的用户：

1. 获取当前用户的ID。

2. 在关注表中检索所有关注者用户ID与当前用户ID对应的记录。

3. 根据关注者用户ID，从用户表中获取相关的用户信息，例如用户名、年龄、学校等。

4. 构建已关注用户的列表或相关用户信息的数据结构。

5. 返回已关注用户的列表或相关用户信息作为输出数据。

3.3.7、函数说明

具体说明模块中的各个函数，包括函数名称及其所在文件，功能，格式，参数，全局变量，局部变量，返回值，算法说明，使用约束等。

A. 函数名称：followUser()

- 文件：follow.py

- 功能：关注其他用户

- 格式：def followUser(target\_user: str) -> str

- 参数：

- target\_user：目标用户的用户名或用户ID（字符串类型）

- 返回值：成功或失败的提示信息（字符串类型）

- 算法说明：

1. 验证target\_user的有效性，包括检查目标用户是否存在和用户输入的格式是否符合要求。

2. 如果target\_user无效，返回错误提示信息。

3. 检查当前用户是否已经关注了目标用户，以避免重复关注。

4. 将关注关系添加到关注表中，将当前用户的ID和目标用户的ID存储为一条记录。

5. 返回成功的提示信息，表示关注操作已成功。

B. 函数名称：unfollowUser()

- 文件：unfollow.py

- 功能：取消关注用户

- 格式：def unfollowUser(target\_user: str) -> str

- 参数：

- target\_user：目标用户的用户名或用户ID（字符串类型）

- 返回值：成功或失败的提示信息（字符串类型）

- 算法说明：

1. 验证target\_user的有效性，包括检查目标用户是否存在和用户输入的格式是否符合要求。

2. 如果target\_user无效，返回错误提示信息。

3. 检查当前用户是否已经关注了目标用户。

4. 如果关注关系存在，将该关注关系从关注表中删除。

5. 返回成功的提示信息，表示取消关注操作已成功。

C. 函数名称：getFollowedUsers()

- 文件：get\_followed\_users.py

- 功能：查看已关注的用户

- 格式：def getFollowedUsers() -> List[User]

- 返回值：已关注用户的列表，每个用户表示为User对象的列表（List[User]类型）

- 算法说明：

1. 获取当前用户的ID。

2. 在关注表中检索所有关注者用户ID与当前用户ID对应的记录。

3. 根据关注者用户ID，从用户表中获取相关的用户信息，例如用户名、年龄、学校等。

4. 构建已关注用户的列表，每个用户表示为User对象，并将其添加到列表中。

5. 返回已关注用户的列表作为输出数据。

注：使用约束

- 在调用这些函数之前，需要确保用户已经登录，并且有权限执行相关操作。

- 函数内部需要进行合适的错误处理，例如处理无效的输入数据、数据库连接错误等。

3.3.8 全局数据结构与该模块的关系

A. 用户表（User table）：

- 功能：存储所有注册用户的个人信息和用户ID。

- 访问方式：通过查询、插入或更新操作来访问用户表，以获取用户信息、验证用户存在性等。

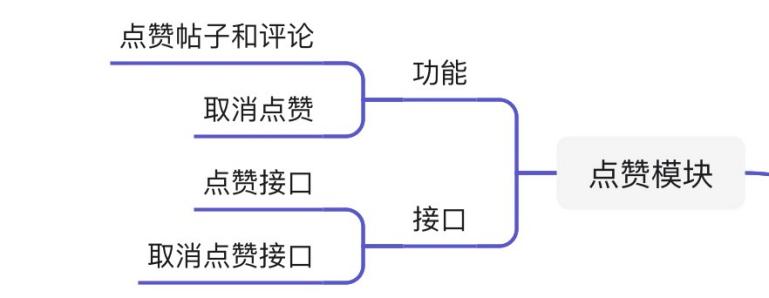
B. 关注表（Following table）：

- 功能：记录用户之间的关注关系，包括关注者用户ID和被关注者用户ID。

- 访问方式：通过查询、插入或删除操作来访问关注表，以建立、取消和检索关注关系。

**3.4、模块2**

3.3.1、设计图



3.3.2、功能描述

点赞帖子和评论、取消点赞

3.3.3、输入数据

1. 用户输入的数据：

- 帖子或评论的ID：用户需要提供要点赞或取消点赞的帖子或评论的唯一标识符，通常是一个ID值。

- 点赞操作类型：用户需要指定是点赞还是取消点赞。

2. 有效性检验规则：

- 帖子或评论的ID有效性检验：系统需要验证提供的帖子或评论ID是否存在于论坛数据库中，确保操作的对象是有效的。如果ID无效，系统会返回相应的错误信息。

- 点赞操作类型有效性检验：系统需要验证用户提供的操作类型是否有效，即只接受"点赞"或"取消点赞"的指令。如果操作类型无效，系统会返回相应的错误信息。

3. 获取数据：

- 帖子和评论数据表：系统通过查询论坛数据库中的帖子和评论数据表来获取相关的数据。通常，这些表会包含帖子或评论的ID、作者信息、内容、点赞数量等字段。

- 获取数据的条件：系统需要根据用户提供的帖子或评论ID，在帖子或评论数据表中进行查询。如果查询结果为空，则意味着提供的ID无效，系统会返回相应的错误信息。

3.3.4、输出数据

1. 点赞记录数据：

- 帖子点赞记录：记录用户对帖子进行点赞或取消点赞的操作记录。

- 数据：用户ID、帖子ID、点赞时间、点赞状态（点赞或取消点赞）等字段。

- 表现形式：这些数据可以以数据库中的表形式存储，每一行代表一个点赞操作记录，不同字段对应表的不同列。

- 评论点赞记录：记录用户对评论进行点赞或取消点赞的操作记录。

- 数据：用户ID、评论ID、点赞时间、点赞状态（点赞或取消点赞）等字段。

- 表现形式：同样以数据库中的表形式存储，每一行代表一个点赞操作记录，不同字段对应表的不同列。

2. 点赞数量数据：

- 帖子点赞数量：记录每个帖子收到的点赞总数。

- 数据：帖子ID、点赞数量等字段。

- 表现形式：可以将这些数据存储在数据库的帖子表中，通过一个字段来表示点赞数量。

- 评论点赞数量：记录每个评论收到的点赞总数。

- 数据：评论ID、点赞数量等字段。

- 表现形式：可以将这些数据存储在数据库的评论表中，通过一个字段来表示点赞数量。

3.3.5、数据设计

给出本程序中的局部数据结构说明，包括数据结构名称，功能说明，具体数据结构说明（定义、注释设计、取值）等。相关数据库表，数据存储设计（具体说明需要以文件方式保存的数据文件名、数据存储格式、数据项及属性等。）

3.3.6、算法和流程

详细描述根据输入数据产生输出数据的算法和流程。

3.3.7、函数说明

具体说明模块中的各个函数，包括函数名称及其所在文件，功能，格式，参数，全局变量，局部变量，返回值，算法说明，使用约束等。

3.3.8 全局数据结构与该模块的关系

说明该模块访问了哪些全局数据结构。

# 4、 接口设计

**4.1、 用户接口**

说明将向用户提供的接口。

**4.2、 外部接口**

前端和后端之间的接口采用RESTful API设计风格，使用HTTP协议进行通信。前端通过axios库向后端发送请求，后端返回JSON格式的响应。

**4.3、 内部接口**

4.3.1、 接口说明

例如：xx子模块通过xx从xx子模块取得xx等，相关标准，调用示例，可根据需要增加章节描述接口。

4.3.2、 调用方式

例如：

/\*\*

\*通过用户服务号码取得该客户认证密码等信息，如果该客户存在返回为0，其他情况参考错误编码

\*/

public RUserInfo getUserInfo (String userNo);

# 5、数据库设计

本系统中使用的数据库包括以下数据表：

关注表（codes）：

标识符：codes

数据项：

id: 验证码ID，整数类型，主键，自增长

code: 验证码，字符串类型，唯一

email: 电子邮箱，字符串类型，唯一

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

email: 电子邮件，字符串类型，唯一

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。

# 6、系统出错处理

**6.1、 出错信息**

用一览表的方式说明每种可能的错误和故障，以及系统输出信息的形式、含义和处理方式。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **可能的错误和故障** | **输出信息的形式** | **输出信息的含义** | **处理方式** |
| 1 | 资源未找到 | 404 Not Found | 请求的资源在服务器上未找到 | 检查请求的 URL 是否正确，确保所请求的资源存在。如果资源不存在，可以返回适当的错误响应或重定向到合适的页面。 |
| 2 | 请求错误 | 400 Bad Request | 客户端发送的请求存在语法错误或无效数据 | 解析请求数据并验证其格式和内容。如果请求无效，可以返回适当的错误响应，包含错误细节和可能的修复建议。 |
| 3 | 未经授权 | 401 Unauthorized | 请求需要进行身份验证，但未提供有效的身份凭证 | 请求需要进行身份验证，但未提供有效的身份凭证。  处理方式：要求用户提供有效的身份凭证，如用户名和密码。如果仍未经授权，返回适当的错误响应或重定向到登录页面。 |
| 4 | 禁止访问 | 403 Forbidden | 请求被服务器理解，但服务器拒绝执行该请求 | 检查请求的权限和访问控制设置。如果用户没有足够的权限，返回适当的错误响应，说明原因或提供其他可行的操作。 |
| 5 | 服务器内部错误 | 500 Internal Server Error | 服务器在处理请求时遇到了意外的错误 | 记录错误详细信息，包括错误类型等，并将其显示或记录在日志中。对于用户，可以返回一个友好的错误消息，表明系统出现了问题，并建议稍后重试或与管理员联系。 |
| 6 | 实体无法处理 | 422 Unprocessable Entity | 请求的数据存在语义错误或无效而验证失败时，FastAPI 会自动生成带有错误详细信息的 422 响应，以告知客户端请求存在问题。 | 根据需求自定义错误处理器，并选择其他错误码来表示不同类型的错误。可以使用 FastAPI 的异常处理机制和自定义异常处理器来捕获和处理验证错误，以及返回适当的错误码和错误信息。 |
| 7 | 数据库错误 | 数据库查询错误、连接超时、写入冲突等 | 与数据库交互时发生错误或异常 | 捕获数据库相关的异常并进行适当处理，如重新连接数据库、重试操作，或返回与数据库错误相关的错误信息。 |
| 8 | 网络故障 | 网络连接中断、DNS 解析失败等 | 与外部服务通信时出现问题 | 检查网络连接、服务可用性和配置。对于无法访问的外部服务，可以返回一个适当的错误响应，指导用户稍后重试或检查网络连接。 |

**6.2、 补救措施**

说明故障出现后可能采取的补救措施，如恢复、再启动技术等。

对于每种错误和故障，将根据实际情况采取适当的处理方式，包括返回适当的错误响应、记录日志、提供用户友好的错误消息、重试操作或与管理员联系等。确保错误和故障信息能够及时捕获、记录和通知，以便及时解决问题并提供良好的用户体验。

# 7、其他设计

**7.1 系统安全设计**

用户认证与授权：使用身份验证机制来验证用户身份，使用OAuth 2.0 和 Bearer JWT来生成和验证访问令牌，并确保只有经过身份验证和授权的用户才能访问受保护的资源。

密码安全：采用适当的密码存储和加密方式，使用哈希函数bcrypt对用户密码进行加密，确保用户密码不以明文形式存储在数据库中。

输入验证与过滤：在前后端都进行输入验证，过滤用户输入的数据，防止恶意输入、SQL注入等攻击。

安全漏洞防护：定期更新框架和依赖库，及时修复安全漏洞。限制对敏感接口和资源的访问。

**7.2 性能设计**

数据库性能优化：使用适当的索引和查询优化来提高数据库查询性能。合理设计数据库表结构，避免数据冗余和不必要的关联。

前端性能优化：使用前端技术和工具来优化页面加载速度。

后端性能优化：使用异步处理来提高请求处理能力。

代码优化：优化代码逻辑，避免不必要的计算和数据库查询操作。使用合适的数据结构和算法来提高代码效率。

异常处理与报警：及时捕获和处理异常，发送报警通知以便及时响应和解决问题。