**目 录**

[一、引言（1-2页） 1](#_Toc170832060)

[1.1 研究背景 1](#_Toc170832061)

[1.1.1行业背景: 1](#_Toc170832062)

[1.1.2技术背景: 1](#_Toc170832063)

[1.2 研究问题 2](#_Toc170832064)

[1.2.1问题识别: 2](#_Toc170832065)

[1.2.2问题重要性: 2](#_Toc170832066)

[1.3 研究目的 2](#_Toc170832067)

[1.3.1目的阐述: 2](#_Toc170832068)

[1.3.2预期目标: 2](#_Toc170832069)

[1.4 研究意义 3](#_Toc170832070)

[1.4.1理论意义: 3](#_Toc170832071)

[1.4.2实践意义: 3](#_Toc170832072)

[1.5 研究范围与限制 4](#_Toc170832073)

[1.5.1研究范围: 4](#_Toc170832074)

[1.5.2研究限制: 4](#_Toc170832075)

[二、技术选型与环境搭建 4](#_Toc170832076)

[2.1 技术栈介绍 4](#_Toc170832077)

[ **2.1.1开发语言**: python3.7.9 4](#_Toc170832078)

[ **2.1.2框架和库**: 4](#_Toc170832079)

[2.2 开发环境配置 7](#_Toc170832080)

[ **2.2.1操作系统**: Windows10、11 7](#_Toc170832081)

[ **2.2.2开发工具**: pycham 7](#_Toc170832082)

[ **2.2.3数据库系统**: 7](#_Toc170832083)

[ **2.2.4服务器配置**: 7](#_Toc170832084)

[2.3 工具和库的选择理由 7](#_Toc170832085)

[ **2.3.1选择标准**: 7](#_Toc170832086)

[ **2.3.2优势分析**: 7](#_Toc170832087)

[2.4 实训环境的特殊要求 8](#_Toc170832088)

[ **2.4.1硬件要求**: 8](#_Toc170832089)

[ **2.4.2网络环境**: 9](#_Toc170832090)

[2.5 环境搭建步骤 9](#_Toc170832091)

[ **2.5.1安装指南**: 9](#_Toc170832092)

[ **2.5.2配置文件**: 10](#_Toc170832093)

[2.6 环境测试 11](#_Toc170832094)

[ **2.6.1测试方法**: 11](#_Toc170832095)

[ **2.6.2测试结果**: 11](#_Toc170832096)

[三、团队合作与分工 12](#_Toc170832097)

[3.1 团队结构 13](#_Toc170832098)

[ **团队规模与角色分配**: 13](#_Toc170832099)

[3.2 分工原则 13](#_Toc170832100)

[3.3 合作机制 13](#_Toc170832101)

[ **3.3.1沟通方式**: 13](#_Toc170832102)

[ **3.3.2决策过程**: 13](#_Toc170832103)

[3.4 任务执行 13](#_Toc170832104)

[3.5 风险管理 14](#_Toc170832105)

[3.6 成员贡献 14](#_Toc170832106)

[3.7 团队反思 14](#_Toc170832107)

[四、项目实现过程 15](#_Toc170832108)

[4.1 需求分析 15](#_Toc170832109)

[4.1.1用户需求: 15](#_Toc170832110)

[4.1.2功能需求: 15](#_Toc170832111)

[4.1.3非功能需求: 16](#_Toc170832112)

[4.2 设计阶段 16](#_Toc170832113)

[4.2.1系统架构: 16](#_Toc170832114)

[4.2.2模块划分: 17](#_Toc170832115)

[4.2.3接口设计: 17](#_Toc170832116)

[4.3 开发准备 18](#_Toc170832117)

[4.3.1环境搭建: 18](#_Toc170832118)

[4.3.2工具选择: 18](#_Toc170832119)

[4.4 编码实现 18](#_Toc170832120)

[4.4.1编码规范: 18](#_Toc170832121)

[4.4.2模块开发: 19](#_Toc170832122)

[4.5 版本控制 21](#_Toc170832123)

[4.5.1版本管理: 21](#_Toc170832124)

[4.5.2代码审查: 21](#_Toc170832125)

[4.6 功能测试 22](#_Toc170832126)

[4.6.1测试计划: 22](#_Toc170832127)

[4.6.2测试执行: 23](#_Toc170832128)

[4.7 问题解决 26](#_Toc170832129)

[4.7.1问题记录: 26](#_Toc170832130)

[4.7.2解决方案: 26](#_Toc170832131)

[4.8 系统集成 27](#_Toc170832132)

[五、项目总结（1-2页） 27](#_Toc170832133)

[附录 28](#_Toc170832134)

# 一、引言（1-2页）

## 1.1 研究背景

### 1.1.1行业背景:

B站（哔哩哔哩，Bilibili）是中国领先的视频分享平台，以其丰富的内容、活跃的社区和独特的文化氛围著称。随着互联网和移动互联网的发展，视频分享平台在全球范围内迅速崛起，成为人们获取信息、娱乐和交流的重要途径。B站作为中国的代表性平台，不仅吸引了大量的用户，还成为二次元文化和年轻人文化的重要传播渠道。内容多元化、社区互动、商业化探索、技术驱动、全球化发展

### 1.1.2技术背景:

大数据与数据分析：数据分析用于用户评论的情感分析、视频内容的统计分析等方面。

人工智能与自然语言处理：AI技术，尤其是自然语言处理（NLP）技术，在评论分析、情感分析、内容理解等方面发挥了重要作用。

云计算与分布式系统：视频分享平台需要处理海量数据和高并发请求，云计算和分布式系统提供了高效的数据存储和处理能力，保证了平台的稳定运行。

Web开发技术：现代Web开发框架（如Flask）和前端技术（如HTML、CSS、JavaScript）为平台的开发和用户界面设计提供了便利。

数据库技术：关系型数据库（如MySQL）在数据存储、查询优化和数据一致性方面提供了坚实的基础。

安全技术：应用安全技术（如HTTPS、数据加密、用户认证）来保护用户数据安全。

## 1.2 研究问题

### 1.2.1问题识别:

评论情感分析研究问题：如何提高情感分析的准确性和可靠性，以便更好地理解用户的情感态度？

数据可视化：如何通过数据可视化技术，使数据展示更加直观、易懂，帮助用户更好地理解数据？

系统的可扩展性和维护性：如何设计系统架构，使其具有良好的可扩展性，并便于后续的功能扩展和系统维护？

### 1.2.2问题重要性:

评论情感分析：挖掘用户的情感倾向和意见，提高情感分析的准确性可以帮助平台更好地理解用户需求，提升用户满意度，优化内容推荐和用户互动。

数据可视化：数据可视化是展示数据的有效方式，可以帮助用户更直观地理解和分析数据。

系统的可扩展性和维护性：系统的可扩展性和维护性直接影响其长期运行的稳定性和易用性。

## 1.3 研究目的

### 1.3.1目的阐述:

本项目旨在构建一个基于 Flask 框架的 Web 应用系统，该系统能够有效地处理用户登录、数据展示和后台管理等功能。主要目的是提供一个平台，让用户可以通过友好的界面进行数据交互，管理和分析视频相关的数据。这包括从数据库中获取用户信息、视频信息、评论信息等，并将这些数据可视化展示给用户。

### 1.3.2预期目标:

实现用户登录与注册功能：

通过表单和数据库交互，实现用户信息的注册、登录和认证。

确保用户密码的安全存储和传输。

提供数据管理界面：

使用 Flask Admin 实现一个功能齐全的后台管理界面，方便管理员查看和管理用户、视频、评论等信息。

提供增删改查功能，确保数据的完整性和一致性。

实现数据的可视化展示：

通过图表和可视化工具，将视频数据、评论数据等直观地展示给用户。

实现数据过滤和分析功能，帮助用户更好地理解数据的趋势和模式。

确保系统安全性和用户体验：

通过 Flask 的 session 机制，确保用户会话的安全性。使用响应式设计，确保系统在各种设备上都有良好的显示效果。

## 1.4 研究意义

### 1.4.1理论意义:

数据驱动的社会分析： 该系统通过分析B站用户的评论数据，可以深入理解用户在观看视频时的态度、情感倾向以及互动模式。这为社会学、传播学等领域提供了一个大规模的社交数据分析案例，有助于理解现代社交媒体平台上用户行为背后的动机和模式。

情感分析与文本挖掘： 通过对评论内容进行情感分析（如利用Snownlp），可以探讨自然语言处理在大数据环境下的应用，尤其是在社交媒体数据挖掘中的效果和限制。这对于情感分析算法的改进以及对用户情感及态度的更深层理解具有重要意义。

用户行为模式与影响力研究： 通过分析视频列表、评论热度等数据，可以揭示用户在B站上的行为模式和影响力。这对于研究用户在在线社区中的影响力传播机制、用户粘性和社会网络结构的形成具有理论上的启发。

### 1.4.2实践意义:

1.该系统提供了一个基于数据驱动的社交媒体管理工具，可以帮助运营团队了解用户偏好、内容热度和社区互动情况，从而优化内容推荐、增强用户黏性，提升平台活跃度。

2.通过用户角色管理功能，可以实现对用户权限的灵活控制，保障平台安全和用户体验。另外，基于用户行为数据的个性化推荐系统可以提升用户满意度和平台的内容精准度。

3. 通过情感分析和评论内容云图等功能，可以实时监测用户对内容的反馈和态度变化，及时发现负面舆情并进行有效处理，维护平台形象和用户信任。

商业决策支持： 数据可视化功能（如评论数据的图表展示）为市场营销团队和内容创作者提供了数据驱动的决策支持工具，帮助优化内容策略、增加广告投放精准度，提升商业价值。

## 1.5 研究范围与限制

### 1.5.1研究范围:

本项目开发了一个基于Flask框架的B站评论分析系统，旨在提供一个稳定、安全且易于使用的平台，用于获取、分析和可视化B站视频的评论数据。研究涵盖的内容包括：

项目架构设计、数据爬取、数据处理和分析、数据可视化、前端开发、用户管理

### 1.5.2研究限制:

数据准确性：由于评论数据来源于网络爬虫，可能存在数据遗漏或不完整的情况，影响分析结果的准确性。

情感分析模型：使用的情感分析模型可能存在一定的误差，特别是对含有复杂语言或网络用语的评论，其准确性可能受到影响。

系统性能：在处理大规模数据时，系统性能可能受到限制，影响响应速度和用户体验。

# 二、技术选型与环境搭建

## 2.1 技术栈介绍

* **2.1.1开发语言**: python3.7.9
* **2.1.2框架和库**:

flask==1.1.4 ：Flask 是一个轻量级的 Web 框架，它基于 Werkzeug WSGI 库和 Jinja2 模板引擎。它使得构建 Web 应用程序变得简单和快速  
Werkzeug==1.0.1 ：Werkzeug 是一个 WSGI 工具库，它提供了用于构建 Web 应用程序的核心功能，如请求和响应对象的处理  
markupsafe==2.0.1 ：MarkupSafe 是一个 HTML、XML 和其他标记语言的字符串安全标记库，用于避免注入攻击和其他安全问题  
Jinja2==2.10.1 ：Jinja2 是一个现代的，功能丰富的模板引擎，专门设计用于 Python 编程语言。它可以生成任何基于文本的格式，但最常见的用途是 HTML 页面的生成  
click==7.1.2 ：Click 是一个 Python 模块，用于创建命令行界面（CLI）。它使得开发命令行工具变得简单和直观  
itsdangerous==1.1.0 ：itsdangerous 是一个用于处理 Web 应用中安全签名数据的 Python 库。它可以用于生成并验证签名，例如在处理用户会话时非常有用，以确保数据在传输过程中不被篡改  
fake-useragent==1.1.3 ：fake-useragent 是一个 Python 库，用于生成随机的 User-Agent 字符串，可以用于爬虫程序或需要模拟浏览器的场景中  
pandas==1.1.4 ：pandas 是一个强大的数据分析工具，提供了快速、灵活和富有表现力的数据结构，特别是用于处理结构化数据或表格数据  
flask\_sqlalchemy==2.5.1 ：flask\_sqlalchemy 是 Flask 框架的一个扩展，简化了在 Flask 应用中使用 SQLAlchemy 进行数据库操作的过程  
flask\_babelex==0.9.4 ：flask\_babelex 是 Flask 框架的一个扩展，用于支持多语言和国际化功能。它简化了在 Flask 应用中管理和翻译多语言文本的过程  
flask\_admin==1.6.1 ：flask\_admin 是一个 Flask 扩展，用于快速构建基于 SQLAlchemy 数据库模型的管理界面。它提供了自动生成的 CRUD 操作界面，方便开发人员管理后台数据  
sqlalchemy==1.4.32 ：sqlalchemy 是 Python 中最流行的 SQL 工具和对象关系映射（ORM）库之一。它提供了灵活且强大的方式来管理数据库，支持多种数据库系统，并且可以与 Flask、Django 等框架集成  
Sphinx==1.8.0 ：Sphinx 是一个文档生成工具，特别用于 Python 项目。它支持从标记文本生成各种格式的文档，如 HTML、PDF 和 ePub，并提供了丰富的扩展和主题  
selenium==3.141.0 ：selenium 是一个自动化测试工具，用于模拟浏览器操作。它可以用于测试 Web 应用程序，进行用户界面测试或者进行网页数据抓取等任务  
bs4==0.0.1 ：bs4 是 BeautifulSoup4 的一个旧版本，用于解析 HTML 和 XML 文档，提供了简单且灵活的方式来提取和操作文档中的数据  
wtforms==3.0.1 ：wtforms 是一个用于构建 Web 表单的灵活库，它允许开发者定义表单字段及其验证规则，并且可以方便地集成到 Flask、Django 等 Web 框架中使用  
jieba ：jieba 是 Python 中一个非常好用的中文分词工具，用于将中文文本分割成词语，是自然语言处理中常用的工具之一  
flask\_script ：flask\_script 是 Flask 扩展之一，提供了一组在命令行中运行脚本的功能，例如启动开发服务器、数据库初始化等任务  
numpy ：numpy 是 Python 中用于科学计算的核心库，提供了强大的多维数组对象和数学函数库，适合用于数据处理和数值计算  
snownlp ：snownlp 是一个中文文本情感分析的库，用于分析文本情感倾向，包括情感分类、情感评分等功能  
pyecharts ：pyecharts 是一个基于 Echarts 的 Python 可视化库，用于生成各种类型的交互式图表，如折线图、柱状图、地图等  
wordcloud ：wordcloud 是一个生成词云的 Python 库，用于根据输入的文本数据生成具有艺术性的词云图像  
pymysql ：pymysql 是一个 Python 对 MySQL 数据库进行操作的库，提供了与 MySQL 数据库交互的各种方法和接口  
pillow ：pillow 是 Python Imaging Library（PIL）的一个分支，用于处理图像文件，包括打开、保存、转换和基本的图像处理操作  
requests ：requests 是 Python 中一个简洁而优雅的 HTTP 库，用于发送 HTTP 请求，支持各种 HTTP 方法和 SSL 证书验证  
gensim ：gensim 是一个用于文本建模和主题建模的库，特别适用于处理大规模文本数据，提供了高效的文本相似度计算和主题挖掘功能  
nltk ：nltk（Natural Language Toolkit）是 Python 中自然语言处理（NLP）的一站式解决方案，提供了丰富的语料库和算法，用于文本处理、标记、词性标注等任务

## 2.2 开发环境配置

* **2.2.1操作系统**: Windows10、11
* **2.2.2开发工具**: pycham
* **2.2.3数据库系统**:

mysql Ver 8.0.37 for Win64 on x86\_64 (MySQL Community Server - GPL)

* **2.2.4服务器配置**:

Nginx、Django、Flask、MySQL

## 2.3 工具和库的选择理由

* **2.3.1选择标准**:

性能：工具和库需要具备高效的性能，能够处理大量数据和请求。

兼容性：确保与项目的整体架构和其他工具的兼容性，避免集成上的障碍。

社区支持：选择有活跃社区支持的工具和库，以便在遇到问题时能获得帮助和参考。

易用性：工具和库应易于学习和使用，降低开发难度，提高开发效率。

扩展性：能够支持项目未来的扩展需求，具备良好的可扩展性。

* **2.3.2优势分析**:

Flask

优势：轻量级：Flask 是一个轻量级的框架，易于理解和使用，适合中小型项目。

灵活性：提供了高度的灵活性，允许开发者根据需求自定义项目结构和功能。

扩展性：有丰富的扩展和插件，能够满足各种扩展需求。

社区支持：拥有活跃的社区支持，提供大量的文档、教程和第三方库。

满足项目需求：Flask 的灵活性和轻量特性非常适合本项目中需要快速开发和迭代的需求，同时提供了足够的扩展能力以支持未来的功能扩展。

SQLAlchemy

优势：ORM 功能：提供了强大的 ORM（对象关系映射）功能，简化了数据库操作。

兼容性：支持多种数据库，包括 MySQL、PostgreSQL、SQLite 等。

查询能力：提供了灵活的查询构建器，支持复杂的数据库查询操作。

社区支持：拥有丰富的文档和活跃的社区支持。

满足项目需求：SQLAlchemy 的 ORM 能力简化了数据库模型的定义和操作，提高了开发效率，同时其灵活的查询功能能够满足项目中复杂数据操作的需求。

Flask-Admin

优势：快速集成：能够快速集成到 Flask 项目中，提供现成的后台管理界面。

可定制性：允许开发者根据需求自定义后台管理界面的功能和样式。

易用性：提供简单易用的 API，降低了后台管理系统的开发难度。

满足项目需求：Flask-Admin 提供了一个开箱即用的后台管理界面，极大地减少了开发时间，同时其定制能力也能满足项目中特殊管理需求。

Jinja2

优势：模板引擎：Jinja2 是一个强大的模板引擎，支持模板继承、宏定义等高级功能。

灵活性：允许在模板中使用 Python 表达式，提供了极大的灵活性。

性能：模板渲染性能高效，能够快速生成 HTML 页面。

满足项目需求：Jinja2 的模板引擎功能使得前后端数据交互和页面渲染变得非常简单和高效，满足项目中动态生成页面的需求。

SnowNLP

优势：中文处理：SnowNLP 专门针对中文文本处理，提供了丰富的功能，如情感分析、分词等。

易用性：简单易用的 API，方便集成到项目中。

满足项目需求：SnowNLP 在中文文本处理方面的强大功能，能够满足项目中对评

论数据进行情感分析和其他文本处理的需求。

## 2.4 实训环境的特殊要求

* **2.4.1硬件要求**:

处理器 (CPU)：

建议：至少双核处理器，推荐四核或以上的现代处理器。

理由：由于 Flask 应用主要是 I/O 密集型任务，适合多线程和并发处理，因此多核处理器可以提升整体性能。

内存 (RAM)：

建议：至少 4GB 内存，推荐 8GB 或以上。

理由：本项目涉及到数据库操作、数据处理和前端渲染，充足的内存能够确保多个进程同时运行，并提升系统响应速度。

存储 (Storage)：

建议：至少 20GB 可用磁盘空间。

理由：存储需求包括操作系统、Python 环境、项目文件、数据库存储以及临时文件和日志等。

显卡 (GPU)：

建议：无特别要求，但如果涉及到大量的数据可视化和机器学习任务，建议使用支持 CUDA 的 NVIDIA GPU。

理由：虽然当前项目主要依赖 CPU 进行数据处理，但在处理更复杂的可视化任务或进行深度学习训练时，GPU 可以显著提升计算效率。

* **2.4.2网络环境**:

带宽 (Bandwidth)：

建议：至少 10 Mbps 的互联网连接。

理由：项目可能需要从外部数据源获取数据、用户需要访问 Web 应用以及管理系统的后台，稳定的网络连接可以确保数据传输的效率和可靠性。

延迟 (Latency)：

建议：网络延迟尽量控制在 50ms 以下。

理由：较低的网络延迟能够提高用户体验，特别是在用户交互频繁和实时数据处理

的场景中。

稳定性：

建议：使用有保障的网络服务提供商，确保网络连接的稳定性和持续性。

理由：网络连接的稳定性对 Web 应用的连续可用性至关重要，特别是在用户访问和数据传输频繁时。

## 2.5 环境搭建步骤

* **2.5.1安装指南**:

以下是详细的安装步骤：

安装操作系统

推荐使用 Ubuntu 20.04 或以上的 Linux 版本，或者 Windows 10、macOS 10.15 或以上。

安装 Python

确保安装 Python 3.8 或以上版本。

可以从 [Python 官网](https://www.python.org/downloads/) 下载并安装。

设置虚拟环境

创建并激活虚拟环境以隔离项目依赖。

python3 -m venv venvsource venv/bin/activate # Linux 或 macOS

.\venv\Scripts\activate # Windows

安装项目依赖

在虚拟环境中，使用 pip 安装项目依赖。

首先解压上传的 flaskproject-b(1).zip 文件，进入项目目录。

然后执行以下命令安装依赖：

pip install -r requirements.txt

安装并配置 MySQL

安装 MySQL 数据库，可以参考 [MySQL 安装指南](https://dev.mysql.com/doc/refman/8.0/en/installing.html)。

创建一个新的数据库：

sql

CREATE DATABASE flaskproject\_db;

配置数据库连接

添加数据库连接配置。

初始化数据库

运行数据库文件

运行项目

启动 Flask 应用

应用将默认运行在 http://127.0.0.1:5000。

* **2.5.2配置文件**:

requirements.txt ：这个文件列出了所有需要安装的 Python 库及其版本。使用 pip install -r requirements.txt 安装依赖

.env ：存储环境变量，如数据库连接字符串和应用的密钥。确保 .env 文件不包含在版本控制中以保护敏感信息

config.py ：配置文件，包含应用的配置信息，如数据库 URI 和其他配置信息。

app.py ：主应用文件，负责初始化 Flask 应用和各种扩展，并定义路由。

models.py ：定义数据库模型，包含用户、视频、评论等数据表的定义。

admin.py ：设置 Flask Admin 后台管理界面。

## 2.6 环境测试

* **2.6.1测试方法**:

功能测试：

用户认证测试：测试用户注册、登录、登出功能，确保用户认证流程正常。

数据管理测试：测试用户、视频、评论的增删改查功能，确保数据库操作正常。

前后端交互测试：测试页面的加载和数据展示，确保前端页面能够正确显示后端数据。

兼容性测试：

跨浏览器测试：在不同浏览器（如 Chrome, Firefox, Safari）中测试页面的显示效果和功能，确保跨浏览器兼容性。

* **2.6.2测试结果**:

功能测试结果：

用户认证测试：所有用户认证功能均正常工作，用户能够成功注册、登录和登出。

数据管理测试：用户、视频、评论的增删改查功能均通过测试，数据库操作正常。

前后端交互测试：页面能够正确加载和显示后端数据，前后端交互正常。

兼容性测试结果：

跨浏览器测试：在 Chrome, edge 中显示效果和功能均正常。

输入验证不足：

问题：部分输入验证未处理，存在潜在的安全风险。

解决方案：用户增加输入验证和过滤。

跨浏览器兼容性问题：

问题：在某些浏览器中，页面样式显示不一致。

解决方案：调整 CSS 样式，确保在不同浏览器中显示一致。

总结

本实训项目采用 Python 3.8 作为主要开发语言，结合 Flask 框架进行 Web 应用开发。开发环境配置在 Ubuntu 20.04 操作系统上，使用 Visual Studio Code 作为集成开发环境（IDE），确保了开发过程的高效和便捷。数据库选择 MySQL 8，因其强大的事务处理能力和对复杂查询的支持。

为了确保项目的可扩展性和性能，服务器配置包括至少双核 CPU、4GB RAM 和 20GB SSD 存储空间。网络环境要求稳定，带宽至少为 10 Mbps，以支持多用户同时访问。

在选择工具和库时，我们考虑了社区的活跃度、文档的完整性以及与现有技术的兼容性。所有选择均经过了严格的评估和测试，以确保它们能够满足项目的需求。

环境搭建步骤详细记录在上文中，包括操作系统的安装、开发工具的配置以及依赖库的安装。我们还进行了一系列的环境测试，以确保所有组件都能正常工作，测试结果表明我们的实训环境稳定可靠，并对遇到的问题进行了有效的解决。

# 三、团队合作与分工

## 3.1 团队结构

## 3.2 分工原则

通过基于技能、兴趣、经验和工作量平衡的分工原则，确保每个团队成员都能在适合自己的领域内发挥最大潜力。这种合理的任务分配不仅提高了工作效率，还增强了团队的凝聚力和协作能力，确保项目能够顺利进行并取得成功。

## 3.3 合作机制

* **3.3.1沟通方式**:

采用面对面小组交流方式针对具体问题进行讨论，如技术难题、功能设计等。

* **3.3.2决策过程**:

团队成员提出各自的观点和建议，进行充分讨论，寻求达成多数成员认同的最佳解决方案。

## 3.4 任务执行

通过使用多种协作工具和明确的进度跟踪方法，高效管理项目任务和确保项目按计划推进。项目组长帮助分配任务和跟踪进度，即时通讯工具确保团队成员之间的高效沟通，文档协作工具帮助管理和共享项目文档，代码质量和测试工具确保代码质量和项目稳定性。综合运用，团队能够高效协作，顺利完成项目任务

## 3.5 风险管理

可能遇到代码漏洞和安全性风险，系统性能和扩展性风险，依赖性风险，项目管理和协作风险。

进行定期小组交流，确保团队成员之间的信息共享和协调。识别潜在的风险并制定相应的应对策略，可以有效地减少项目中的不确定性，提高系统的安全性、性能和稳定性，并确保项目按计划顺利进行。

## 3.6 成员贡献

## 3.7 团队反思

为明确的任务分工和角色分配: 我们的团队由七名成员组成，每位成员根据其专业技能和兴趣被分配了特定的角色。团队中包括项目组长、后端开发者、前端开发者、数据库管理员和测试工程师。分工原则基于每个成员的技术背景和个人偏好，以确保每个人都能在其最擅长的领域发挥最大的潜力。

为了促进有效的沟通和协作，面对面进行小组交，流在风险管理方面，定期举行小组讨论识别潜在的问题并制定相应的应对策略。然后团队调整计划以适应不断变化的需求。

每位成员都对项目的成功做出了重要贡献。例如，项目组长负责协调团队工作和项目设计，前端开发者负责实现界面，后端开发者负责服务器端逻辑的编写，数据库管理员负责数据模型的设计和优化，测试工程师负责测试和确保代码质量。

通过这次项目的实践，认识到团队合作的重要性，并学会了如何在压力下有效沟通和协作。计划在未来的项目中继续改进我们的沟通流程和协作机制，以提高团队效率。每位成员都对项目的成功做出了重要贡献，我们将继续努力，提升团队整体的工作效率和项目质量。

# 四、项目实现过程

## 4.1 需求分析

### 4.1.1用户需求:

我们的目标用户大致包括B站数据分析师、B站视频创作者、媒体分析团队、市场营销人员、学术研究人员等。为了满足目标用户的需求，我们需要全面了解B站的用户行为数据和趋势，以支持决策和策略制定。可以通过分析自己视频的表现和观众反馈，优化内容和提升影响力。需要准确捕捉B站平台上的热点话题和舆论趋势，为新闻报道和内容制作提供参考。需要通过分析用户喜好和行为模式，制定精准的营销策略和推广方案以及需要可靠的数据支持和分析工具。

### 4.1.2功能需求:

1.数据概况展示：

展示B站的整体数据概况，包括热门视频、热门作者、热门评论等。

使用Echarts等可视化工具展示图表，支持用户快速理解和导出数据。

2.评论弹幕分析：

提取评论和弹幕中的关键词、情感倾向等信息。

对特定视频或作者的评论弹幕进行深度分析，揭示用户观点和情感。

3.作者分析：

统计和分析B站的作者，包括视频数量、播放量、粉丝数等数据。

根据发布频率、发布时间等信息，提供作者的优化建议。

4.视频可视化分析：

抓取和分析视频的播放量、点赞量、分享量等数据。

通过可视化图表展示视频的表现情况，支持内容优化和推广决策。

5.舆情分析：

识别热点话题和舆论趋势，通过评论、弹幕等数据分析用户对话题或事件的看法和态度。

6.标题词云图功能：

分析视频标题、评论等，生成词云图展示热门话题和关键词分布情况。

7.注册登录界面和后台数据管理：

用户通过注册登录界面进入系统，管理个人数据和设置。

系统管理员通过后台管理系统进行系统维护和数据管理，确保系统安全和稳定运行。

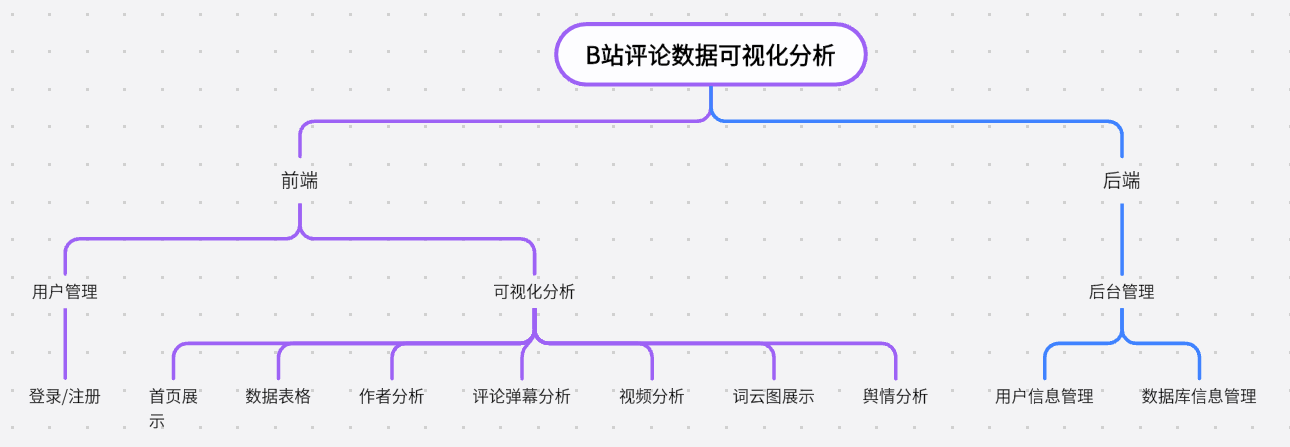
### 4.1.3非功能需求:

为了用户的良好体验，我们的系统还应具有以下非功能需求：

系统响应速度快，支持大规模数据处理和复杂计算。支持高并发访问，保证用户体验流畅。用户数据加密存储和传输，确保数据安全性和隐私保护。防范SQL注入、XSS等安全攻击，保护系统免受恶意攻击。界面友好，操作简单直观，用户易于上手。提供详细的帮助文档和技术支持，解决用户使用中的问题和疑虑。系统架构和代码设计具有良好的扩展性，方便未来功能的添加和升级。系统运行稳定，保证长时间的高效运行，最小化系统故障和宕机时间。支持多种主流浏览器和操作系统，确保用户在不同环境下都能正常使用系统功能。

## 4.2 设计阶段

### 4.2.1系统架构:



### 4.2.2模块划分:

该项目整体被划分为两个模块，即前端和后端，前端包含用户管理以及可视化分析，后端包含后台信息管理。在前端的用户管理中，用户可进行个人账户的登录和注册功能，用户根据登录引导之后进入模块化的可视化分析功能，在此功能模块中，用户可以浏览首页、数据表格、作者分析、评论弹幕分析、视频分析、词云图展示以及舆情分析。在后端的管理模块中，管理员可以对前端的用户信息进行管理，比如删除信息。也可以对前端展示的可视化分析的数据库进行相应的信息修改。

### 4.2.3接口设计:

描述系统各模块之间的接口和交互方式。

该系统通过HTML界面进行各模块之间的跳转，前后端之间的数据传输通过MySQL数据库进行链接。

1、前端用户管理与可视化分析模块之间的接口和交互方式：  
用户通过前端页面进行登录或注册操作，输入用户名和密码进行身份验证，并向后端发送登录或注册请求。

用户登录成功后，前端会向后端发送请求，获取用户的个人信息和权限。

在可视化分析模块中，用户与前端页面交互，浏览不同的功能页面，如数据表格、作者分析等，前端页面会向后端发送相应请求，获取数据进行展示。

用户在进行某些操作时，如设置筛选条件，前端会将用户操作传递给后端，后端进行相应的处理并返回结果给前端展示。

2、后端信息管理模块与前端模块之间的接口和交互方式：  
管理员通过后端管理页面对用户信息进行管理，可以查看用户列表、删除用户等操作，这些操作会触发后端对数据库进行查询或更新操作。

后端还负责管理可视化模块所需的数据库信息，比如视频信息、评论数据、弹幕信息等。

后端还会根据前端的请求（如修改数据库信息、删除用户等）做相应的响应，将更新后的信息返回给前端进行展示或刷新。

## 4.3 开发准备

### 4.3.1环境搭建:

1.安装 PyCharm：

前往 JetBrains 官网下载 PyCharm 的适当版本。按照安装向导完成安装过程。

2.设置 Virtualenv虚拟环境：

在 PyCharm 中创建一个新的项目或打开现有项目。打开项目后，点击顶部菜单栏中的 File -> Settings。在弹出的设置窗口中，展开 Project: YourProjectName，选择 Python Interpreter。点击右上角的齿轮图标，选择 Add... 添加一个新的解释器。在弹出的窗口中选择 Virtualenv Environment，然后选择一个合适的 Python 解释器版本并确定位置。点击 OK 完成设置。我们选择了相同版本的 Python 解释器。

3.安装 MySQL：

前往 MySQL 官网下载适合你操作系统的 MySQL Community Server 版本。按照安装向导进行安装。在安装过程中设置 MySQL 的 root 用户密码，并记住这些凭据。

### 4.3.2工具选择:

1.PyCharm：

理由： PyCharm 是一个功能强大的 Python 集成开发环境（IDE），提供了丰富的功能和插件支持，能够有效地提高开发效率，包括代码提示、调试功能、版本控制等。

2.MySQL：

理由： MySQL 是一个广泛使用的开源关系型数据库管理系统，具有稳定性高、性能优越以及开放源代码等特点，适合用于开发各种规模的应用程序的数据库存储需求。

## 4.4 编码实现

### 4.4.1编码规范:

1.PEP 8: Python Enhancement Proposal 8 是 Python 社区广泛接受的代码风格指南。主要包括：

缩进：使用 4 个空格作为一个缩进层级。

行长：每行代码不超过 79 个字符，注释和文档字符串不超过 72 个字符。

命名规范：变量名小写，用下划线分隔；类名使用驼峰命名法。

空格：操作符周围使用空格；逗号后面使用空格。

注释：清晰简洁的注释，帮助理解代码意图。

2.命名约定：

模块名：小写字母，可以使用下划线。

类名：驼峰命名法（每个单词首字母大写，无下划线）。

函数和变量名：小写字母，用下划线分隔单词。

3.文档字符串：

函数和方法应包含文档字符串（docstring），描述其功能和参数信息。

使用三重双引号 """ 开始和结束文档字符串。

4.导入规范：

每个导入应该独占一行。

标准库导入放在非标准库导入之前。

5.代码组织：

使用适当的空行分隔函数、类定义和方法。

避免代码行过长，适当使用括号和换行符。

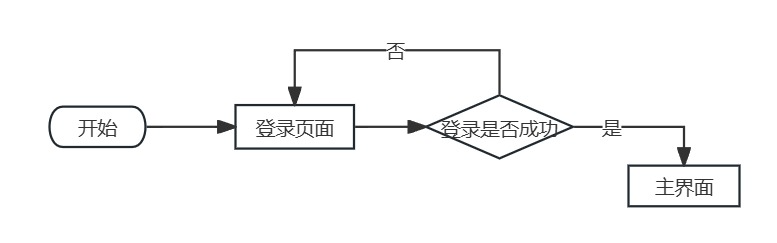
6.版本控制和团队协作：

使用版本控制系统，遵循合适的分支管理和代码审查流程。在团队中建立共识和培养良好的代码审查习惯，确保代码符合标准和规范。

### 4.4.2模块开发:

1.登录模块：

设计流程：

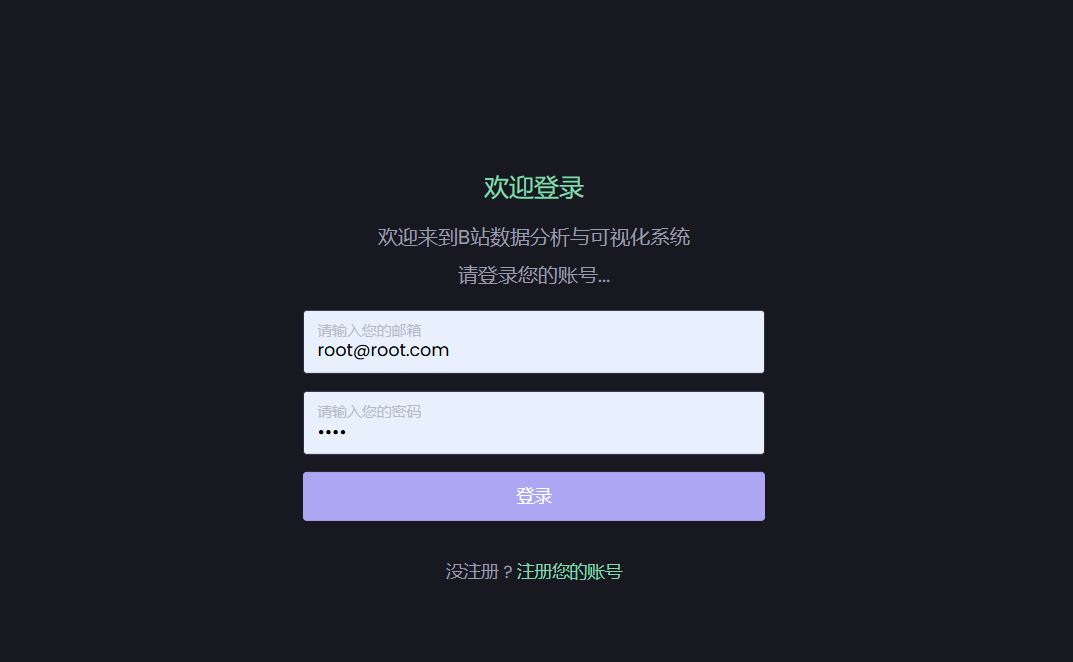


后端设计：

使用 @app.route 装饰器创建了一个路由 /login，支持 GET 和 POST 请求方法。GET 请求将渲染登录页面，POST 请求将处理用户提交的登录表单。检查请求方法是否为 POST，表示用户提交了登录表单。将 request.form 转换为字典形式，方便后续处理和访问表单字段。定义过滤函数和查询用户信息,如果 login\_success 列表为空，说明没有找到匹配的用户，返回错误消息 '账号或密码错误'。如果登录成功，将用户的邮箱存储在会话 session 中，同时将用户角色存储在 userRole 中。这里假设 login\_success[0][-1] 是用户角色信息。最后，使用 redirect 函数将用户重定向到 /home 页面，使用 HTTP 状态码 301。如果请求方法是 GET，直接渲染显示登录页面，使用 render\_template 函数加载 login.html 模板文件。

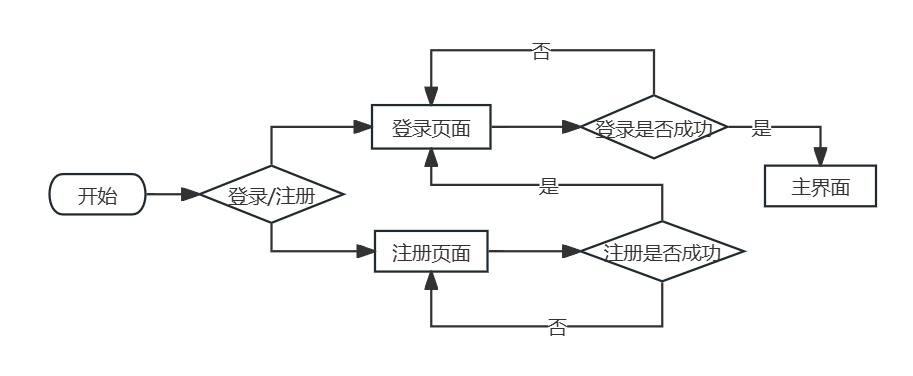
前端设计:

首先通过网上的模板修改确定文件的主题结构，对其进行相关内容修改，然后定义了提交用户登录信息的表单，action="/login"指定了表单提交的地址和方法（POST方式）。表单中包含输入邮箱和密码的字段，分别用<input>标签定义，类型分别为email和password，采用了Bootstrap的form-control类进行样式设置。为了提高用户体验，使用了Bootstrap的form-floating类来实现浮动标签的效果。在页面下方提供了一个注册链接，点击可以跳转到注册页面。引入了多个JavaScript文件，用于页面的交互和动态效果。

前端展示：  


2.注册模块：

设计流程：



后端设计：

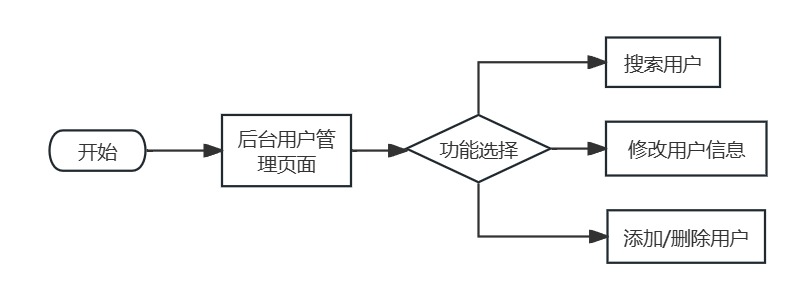
首先当收到 POST 请求时，表示用户正在提交注册信息。首先，将 从表单中接受的数据request.form 转换为字典，方便后续处理。然后，检查用户输入的两次密码是否一致，如果不一致则返回提示信息 '两次密码不符'。如果密码验证通过，接着定义一个 filter\_fn 函数，用于检查用户提交的邮箱是否已经存在于数据库中。通过调用 querys 函数执行 SQL 查询，获取所有用户数据。使用 filter 函数和 filter\_fn 来过滤出邮箱已存在的用户。如果 filter\_list 列表非空，表示该邮箱已经被注册过，返回 '该用户名已被注册' 的提示信息。如果邮箱未被注册，那么执行 SQL 插入语句，将新用户的邮箱和密码插入到数据库中。注册成功后，设置会话信息，将用户的邮箱存入 session 中，以便后续登录时使用。最后，将用户重定向到登录页面 /login，使用 HTTP 状态码 301（永久重定向）。

前端设计：  
 首先通过网上的模板修改确定文件的主题结构，对其进行相关内容修改，注册表单的部分是通过HTML表单元素构建的，<form>标签指定了表单的提交地址/registry，提交方式为POST。这意味着用户在填写完邮箱、密码和确认密码后，点击注册按钮即可向服务器提交注册信息。每个输入字段使用了Bootstrap的.form-floating类，以实现浮动标签效果。例如，邮箱字段使用了<input type="email">，密码和确认密码字段分别使用了<input type="password">。这些字段通过name属性命名，以便在后端处理表单数据时能够准确识别。页面底部提供了一个登录链接，指向/login，让已有账户的用户能够快速跳转到登录页面进行登录操作。最后，为增强页面交互和动态效果，引入了多个JavaScript文件（如jQuery、Bootstrap和自定义脚本），用于处理表单验证、页面动画等功能，提升用户体验和页面的整体性能。

前端展示：  


3.用户信息管理模块

设计流程：



后端设计：

包含用户添加删除修改。

添加功能：首先从 session 中获取当前用户的邮箱 (email) 和用户角色 (userRole)。调用 getPublicData.getAllUpInfo() 函数获取所有的公共信息列表 (upInfoAllList)。假设为 POST 请求，表示用户提交了添加新用户管理者的表单数据。从请求的表单中获取新用户的邮箱 (newEmail) 和密码 (newPassword)。执行数据库插入操作，将新用户的邮箱、密码以及角色信息 (admin) 插入到 user 表中。

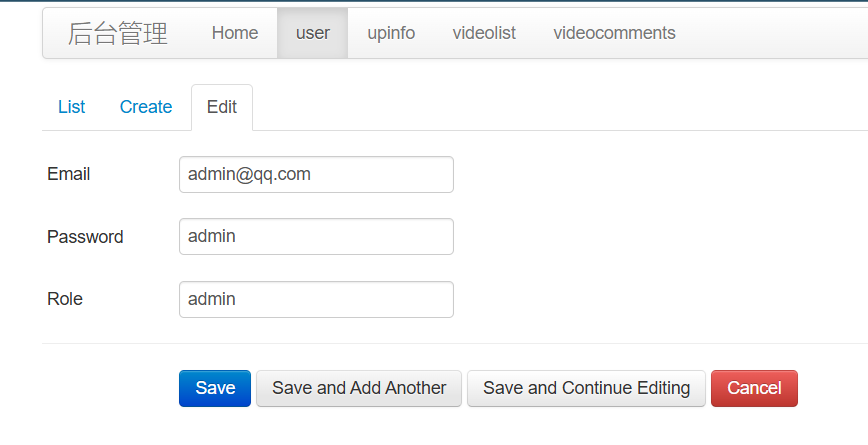
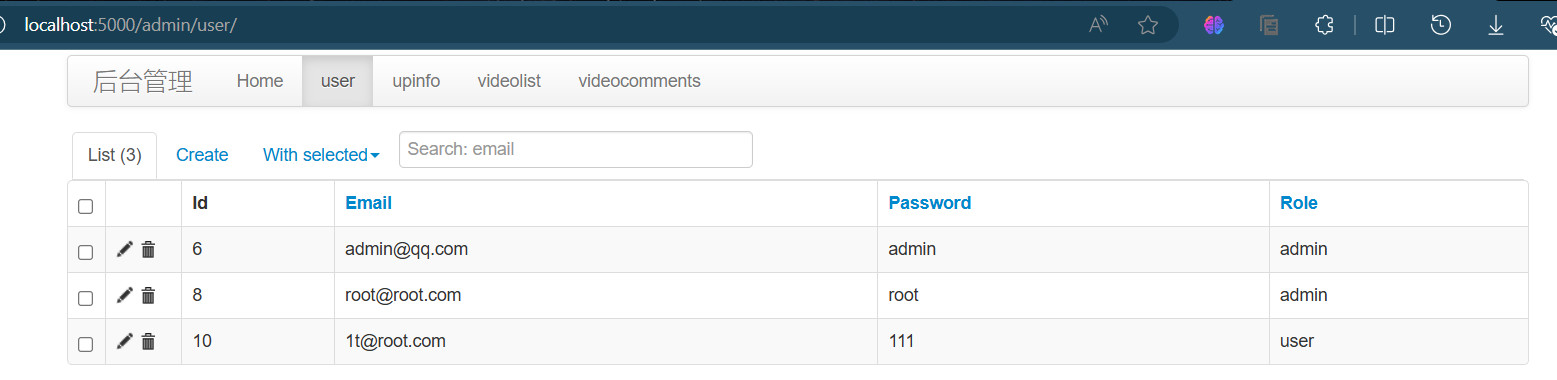
删除功能：调用utils文件夹中预先设定的querys函数查询输入的用户，然后执行数据库删除操作

修改功能：首先从会话中获取当前用户的邮箱和角色信息，以及获取所有UP主信息列表用于页面显示。如果是 GET 请求，根据 userId 查询特定用户的信息，并将其传递给模板进行显示。如果是 POST 请求，从表单中获取新的邮箱和密码，并将其更新到数据库中特定用户的记录。无论是获取用户信息还是提交修改后的信息，最终都通过重定向将用户带回到用户管理页面，提升用户体验和页面流畅度。

前端设计：

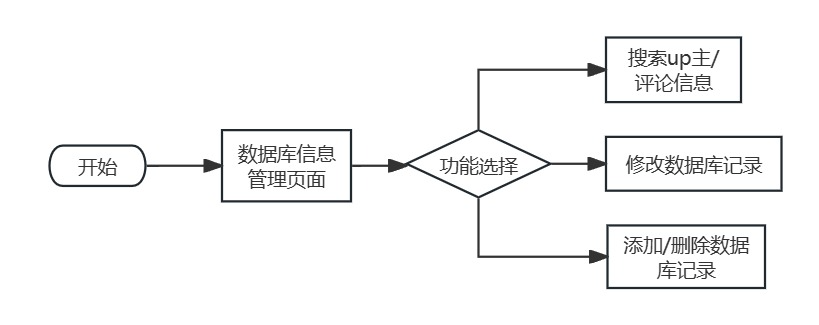
设计#sidebar-toggle 用于切换侧边栏的链接或按钮。header-title 显示了页面标题 "管理员管理"。header-search 包含一个搜索表单，用于搜索 UP 主。用户邮箱和头像显示在导航栏右侧，点击邮箱可以展开退出登录选项。使用了表格 (<table id="zero-conf" class="table table-striped " style="width:100%">) 来展示管理员的信息。表头包括编号、用户名、密码、权限和操作列。使用了模板语法 {% for i in users %} ... {% endfor %} 来循环渲染管理员列表中的每一行数据。每一行的操作列包括了 "修改" 和 "删除" 两个按钮，分别链接到编辑和删除特定管理员账号的页面。页面顶部右侧有一个 "添加管理员账号" 的按钮，点击后跳到 /addUserManager 页面，用于添加新的管理员账号。使用了 Bootstrap 的网格系(<divclass="col-md-4">)，将每个搜索结果以三列的形式展示在页面上。使用模板语法 {% for i inupInfoList %} ... {% endfor %} 遍历服务器传来的 upInfoList 数据，展示每个 UP 主的信息。每个 UP主信息包括了头像、UP 主名称和粉丝数，点击头像或者名称可以跳转到对应 UP 主的详细页面。通过JavaScript渲染功能。

前端展示：

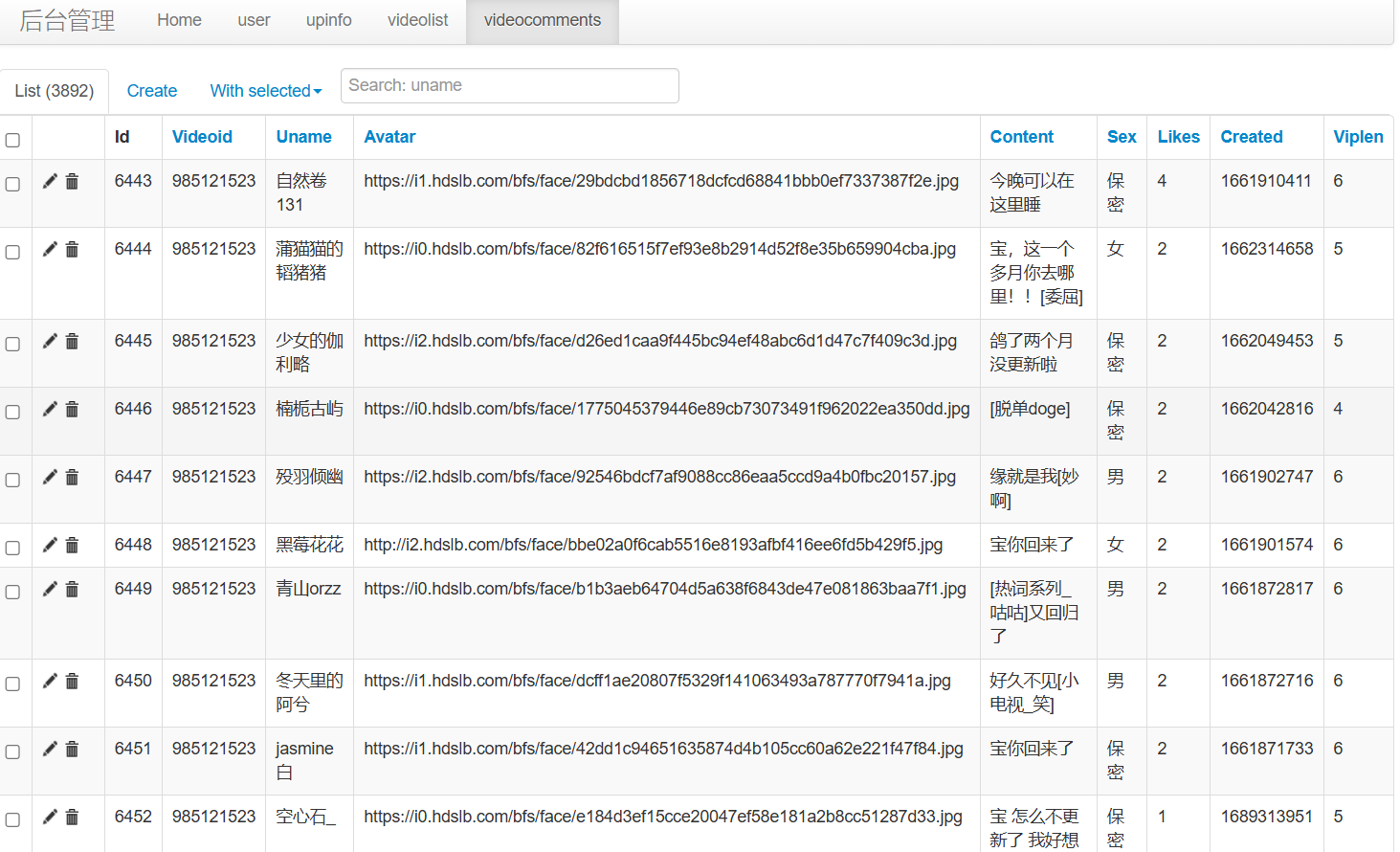


4.数据库信息管理模块

设计流程：

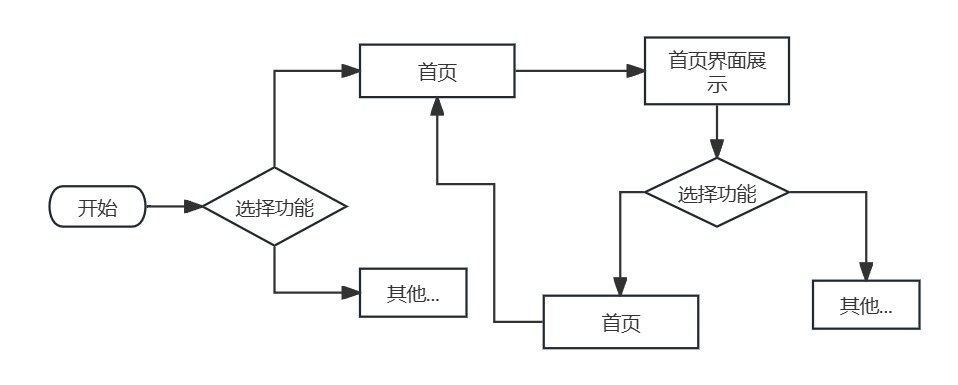


通过关系对象映射的方式将数据库与对象建立联系，然后在通过将数据库模型添加到一个 Flask 后台管理界面中，使得管理员可以通过这个界面管理用户、UP 主信息、视频列表和视频评论等数据。将每个模型视图类注册到后台管理界面中，使得管理员可以通过后台管理界面对这些数据进行管理和操作。每个模型视图类通常提供了诸如列表显示、过滤、搜索、排序、编辑、删除等功能，以便管理员可以方便地管理应用程序中的数据。

前端展示：  


5.首页展示模块

流程：



后端设计：

定义了一个路由处理函数，当用户访问 "/home" 路径时，Flask 将调用这个函数来处理请求。从会话（session）中获取用户的电子邮件地址和用户角色。会话在 Flask 中用于跟踪用户的状态，这些信息通常是用户登录后设置的。通过函数调用调用放在utils中的辅助函数和工具函数获取首页需要展示的各种数据，如首页柱状图数据 getHomeBarData()，页面数据统计getPageData()，首页列表数据 (getHomeList())，以及所有 UP 主信息 getAllUpInfo()。最后，使用 render\_template 函数将获取的数据传递给名为 home.html 的模板文件。模板文件将使用这些数据来动态生成 HTML 页面，向用户展示首页的各种信息，如用户信息、统计数据、列表信息等。

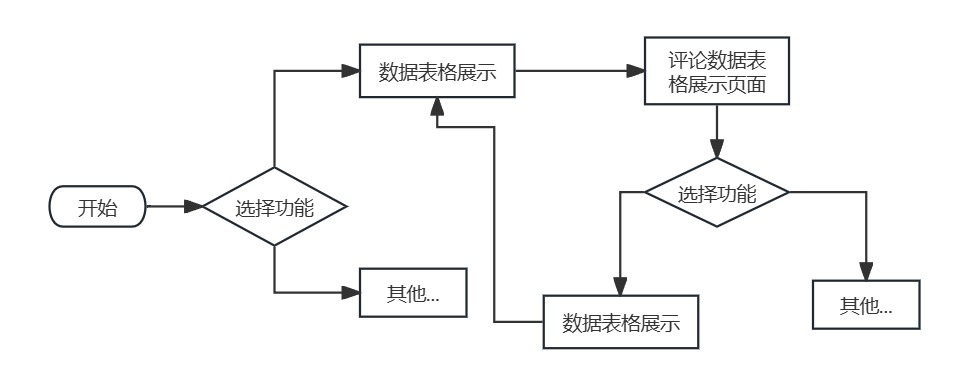
前端设计：

首先，从会话中获取用户的邮箱（email）和用户角色（userRole）。这些数据可能用于显示当前登录用户的信息或控制页面的访问权限。调用getHomePageData中的函数获取各种数据，比如图表数据（xData、yData、y1Data）、最大视频长度相关数据（maxVideoLenName、maxVideoLen）、用户长度数据（userLen）、最大等级数据（maxLevelName、maxLevel）、最大喜欢数数据（maxLikesName、maxLikes）、主播信息列表（upInfoList）、评论最多视频列表（videoListCommentTop）、最受欢迎视频列表（videoListTop）以及所有主播信息列表（upInfoAllList）。在模板中，数据通过对应的变量名（如email、xData、userRole等）进行引用，用于动态生成页面内容。通过模板引擎的语法，将后端传递的数据嵌入到HTML模板中。使用{{ email }}来显示用户邮箱。使用JavaScript渲染xData、yData、y1Data，以及显示最受欢迎视频、评论最多视频、主播信息等列表内容。

前端展示：  


6.数据表格模块

设计流程：



后端设计：

首先定义了一个路由 /tableData，当用户访问这个路径时，Flask 将会调用 tableData() 函数来处理请求。通过 session 对象，获取了用户的登录邮箱和用户角色。这表明该应用程序使用了会话（session）来管理用户状态和身份验证信息。通过调用utils文件夹中的getAllVideoCommentsList() 函数获取了所有视频评论的列表，然后取前 50 个评论 。getAllUpInfo() 方法获取了所有“Up主”的信息列表。最后，使用 render\_template() 函数将数据传递给名为 tableData.html 的模板文件进行渲染。将这些变量userRole、email、commentList、upInfoAllList在模板中被引用，用于动态生成网页内容。

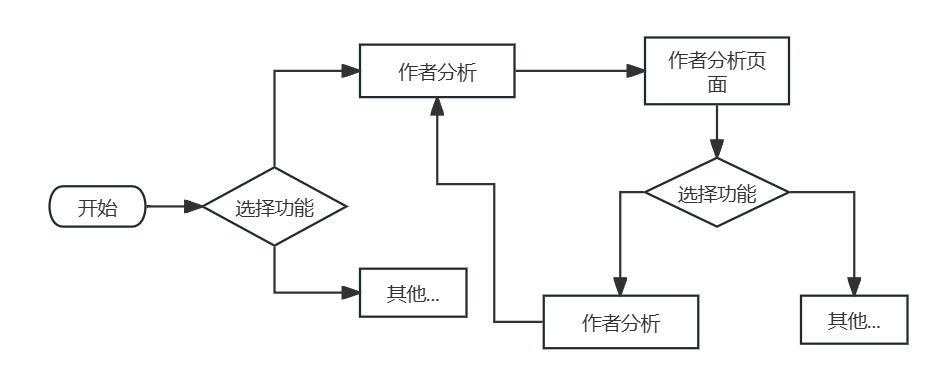
前端设计：

页面使用了Bootstrap框架，结构清晰地分为导航栏、主体内容和页脚部分。将用户的邮箱地址和用户角色这些信息通过后端传递给模板。使用了一个名zero-conf表格插件，具有全局搜索、排序和分页等功能。表格的列定义包括评论的编号、视频ID、评论用户、用户头像（以图片形式展示）、评论内容、用户性别、点赞个数、创建时间和用户等级。通过search 事件处理函数，当用户输入内容时触发。它通过滤 upInfoList 中包含用户输入的部分来实现动态搜索功能。使用了模板语言中的循环语句{% for i in commentList %}来动态地生成表格的每一行，commentList 是一个从后端传递过来的评论数据列表。通过JavaScript代码实现时间的动态更新和本地化显示。

前端展示：  


7.作者分析模块

设计流程：



后端设计：

从 session 中获取当前用户的邮箱 (email) 和用户角色 (userRole)。调用 getPublicData.getAllUpInfo() 获取所有的公共信息列表 (upInfoAllList)，用于显示或者后续操作。调用 getPublicData.getAllVideoList() 获取所有视频列表 (videoAllList)。调用 getPublicData.getAllVideoCommentsList() 获取所有视频评论列表 (videAllCommentVideo)。如果请求的 URL 中包含 upId 参数 (request.args.get('upId'))，则更新 upId 变量为该值。如果请求的 URL 中包含 videoId 参数(request.args.get('videoId'))，则更新 videoId 变量为该值。根据 upId 过滤出特定的 UP 主信息 (upInfo)，并存储在 upInfo 变量中。根据 videoId 过滤出特定的视频信息 (videoInfo)，并存储在 videoInfo 变量中。使用 filter 函数和 lambda 表达式从 videoAllList 和 videAllCommentVideo 中过滤出与当前 upId 和 videoId 相关的数据。将获取到的数据传递给 render\_template 函数，以在页面上显示 UP 主信息 (upInfo)、视频信息 (videoInfo) 和评论信息 (commentInfo)。将 upInfoAllList、videoAllList、commentInfo 等数据列表分别传递给模板，可能用于显示其他相关信息或提供交互选项。最后，将渲染模板的结果返回，呈现给用户查看和交互。

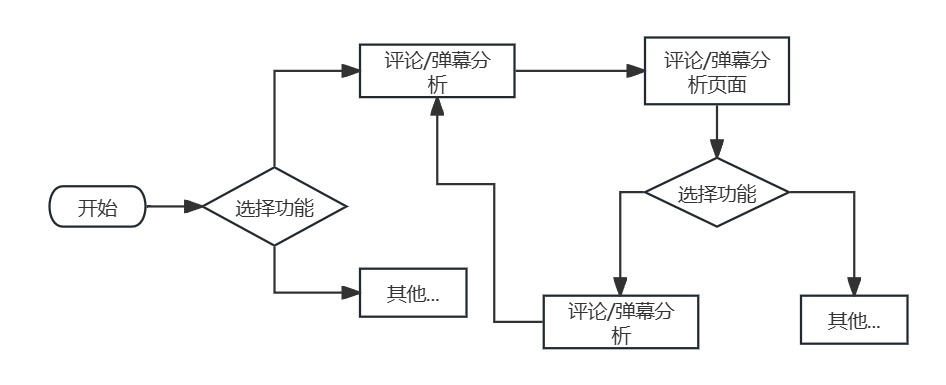
前端设计：

在导航栏设计一个标题和一个搜索框，以及用户个人信息的下拉菜单，显示用户邮箱和头像。包含一个链接或按钮 (#sidebar-toggle) 用于切换侧边栏的显示与隐藏。显示页面标题 "首页"。包含一个搜索框 (header-search)，用于搜索 UP 主。右侧是用户个人信息，包括用户邮箱和头像，点击头像弹出下拉菜单，菜单中有退出登录的选项。通过profile-cover 和 profile-header 分别显示 UP 主的封面和头像，以及 UP 主的名称 (upInfo[2])。card 包含了关于 UP 主的详细信息，如等级、视频数量和粉丝量。显示 UP 主的简介 (upInfo[-1])。使用了无序列表 (list-unstyled profile-about-list) 来展示具体的统计信息，如等级、视频数量和粉丝量。页面中使用了 {{ upInfo[4] }}、{{ upInfo[2] }}、{{ upInfo[-1] }}、{{ upInfo[5] }}、{{ upInfo[6] }} 和 {{ upInfo[3] }} 等模板标记，这些通常是通过后端传递的 UP 主信息来动态填充页面内容的变量。最后设计了一个基于服务器端传来的 UP 主信息列表，实现了简单的搜索功能和结果展示。用户输入内容时，即时更新页面显示符合条件的 UP 主信息，提供了良好的用户体验和交互性。

前端展示：  


8.评论弹幕分析模块

设计流程：



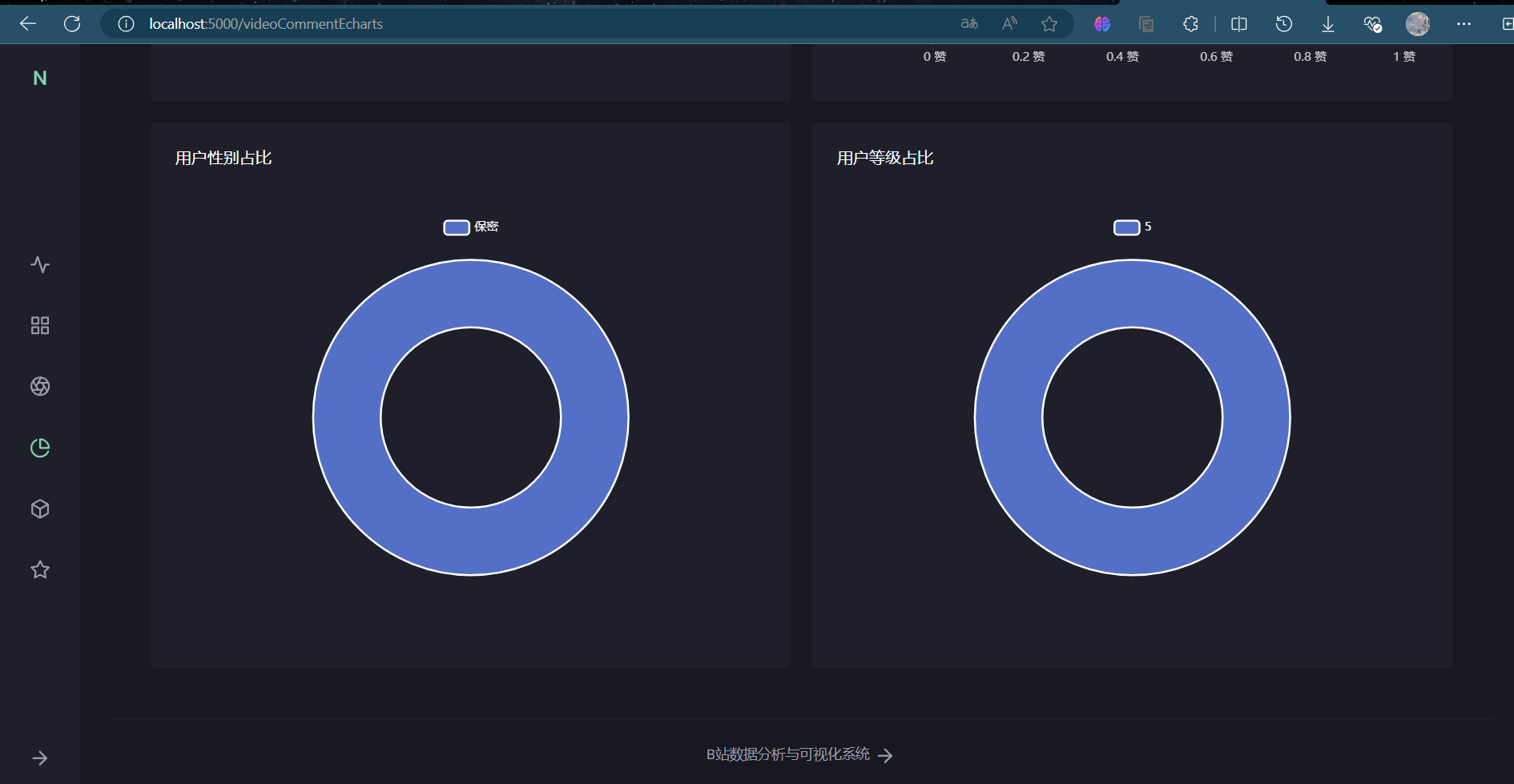
后端设计：

定义了一个路由处理函数，当用户访问 "/videoCommentEcharts" 路径时，Flask 将调用这个函数来处理请求。从会话（session）中获取用户角色。根据用户角色显示页面边栏。通过函数调用调用放在utils中的辅助函数和工具函数获取首页需要展示的各种数据，如所有UP主的信息getAllUpInfo()、视频评论长度相关的数据getCommentLen()、用户评论点赞排名相关的数据getUserCommentLikeTopData()、用户等级分布数据getUserLevelData()最后，使用 render\_template 函数将获取的数据传递给名为 videoCommentEcharts.html 的模板文件。模板文件将使用这些数据来动态生成 HTML 页面，向用户展示首页的各种信息，如用户信息、统计数据、列表信息等。

前端设计：

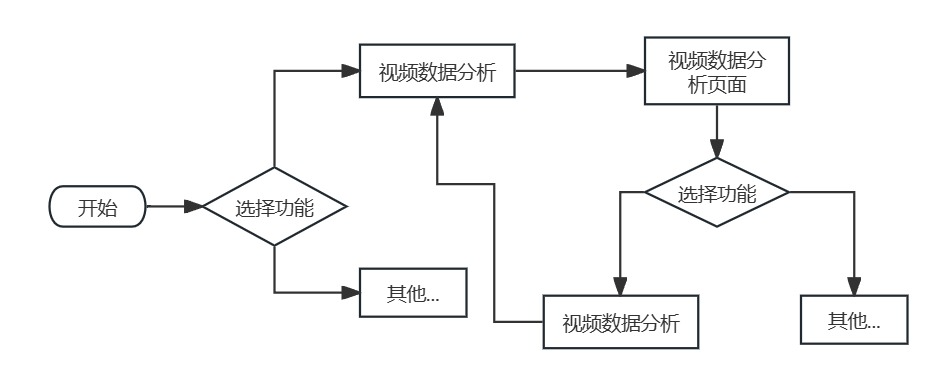
首先在meta标签中设置页面的字符编码、页面的作者信息，在title标签中设置页面的标题为“评论弹幕分析”，利用link标签导入不同的CSS文件用于渲染模板，使用CSS的 border 属性来定义和控制页面边框的样式、宽度和颜色，使用ECharts生成视频评论/弹幕个数柱状图、用户评论点赞Top表柱状图、用户性别占比环状图、

用户等级占比环状图等图形，最终完成评论弹幕分析页面的实现。

前端展示：  


9.视频分析模块

设计流程：

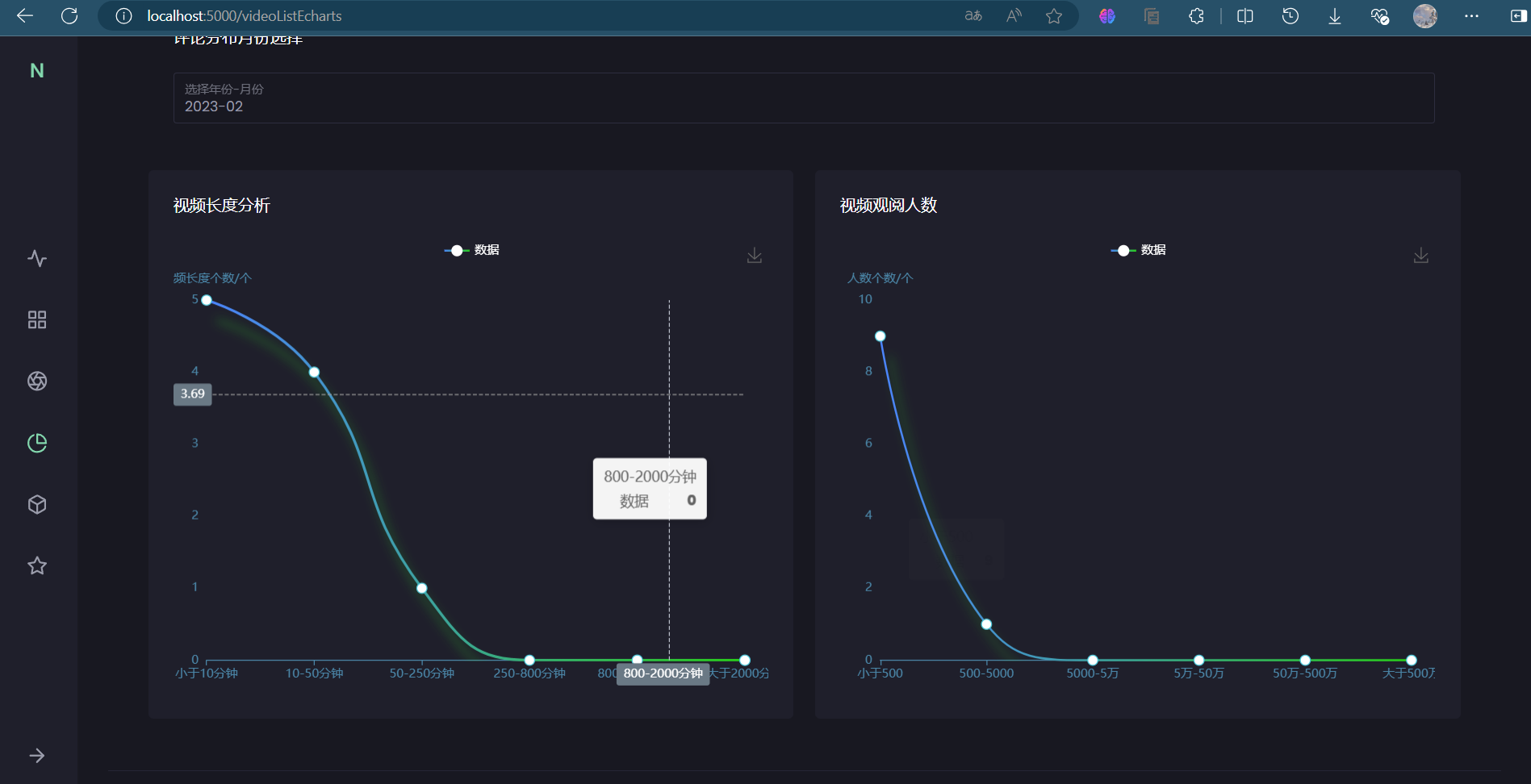


后端设计：

定义了一个路由处理函数，当用户访问 "/videoListEcharts" 路径时，Flask 将调用这个函数来处理请求。从会话（session）中获取用户用户角色。通过函数调用调用放在utils中的辅助函数和工具函数获取首页需要展示的各种数据，如视频评论长度相关的数据getCommentLen()、视频观阅人数数据getSeeNumData()。最后，使用 render\_template 函数将获取的数据传递给名为 videoListEcharts.html 的模板文件。模板文件将使用这些数据来动态生成 HTML 页面，向用户展示视频分析页面的各种信息。

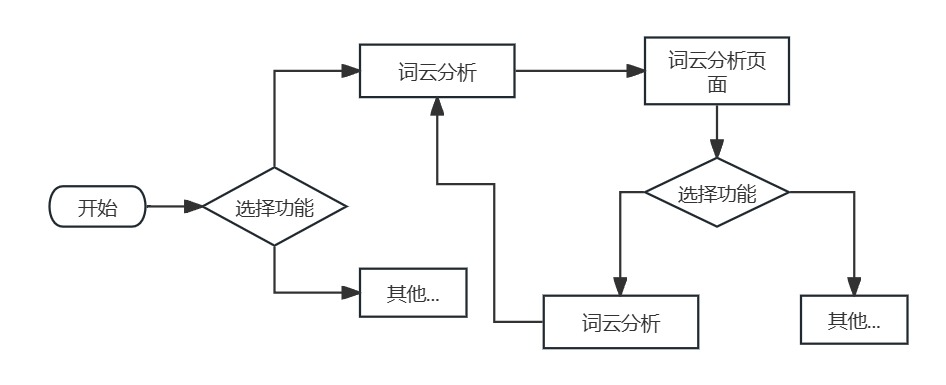
前端设计：

首先在meta标签中设置页面的字符编码、页面的作者信息，在title标签中设置页面的标题为“视频评论分析”，利用link标签导入不同的CSS文件用于渲染模板，使用CSS的 border 属性来定义和控制页面边框的样式、宽度和颜色，使用ECharts生成视频长度分析曲线图、视频观阅人数曲线等图形，最终完成视频分析页面的实现。

前端展示：  


10.词云图模块

设计流程：



后端设计：

定义了两个路由处理函数，当用户访问'/title\_cloud'路径时，该函数被调用。

从会话(session)中获取用户的邮箱地址(email)和用户角色(userRole)。

使用getPublicData模块中的getAllUpInfo()函数获取所有上行信息的列表(upInfoAllList)。

最后，将这些数据传递给名为title\_cloud.html的模板，以便在模板中动态生成页面内容。

当用户访问'/comment\_content\_cloud'路径时，该函数被调用。

与前一个函数类似，从会话中获取用户的邮箱地址(email)和用户角色(userRole)。

同样地，使用getPublicData模块中的getAllUpInfo()函数获取所有上行信息的列表(upInfoAllList)。

将这些数据传递给名为comment\_content\_cloud.html的模板，以便在模板中动态生成页面内容。

前端设计：

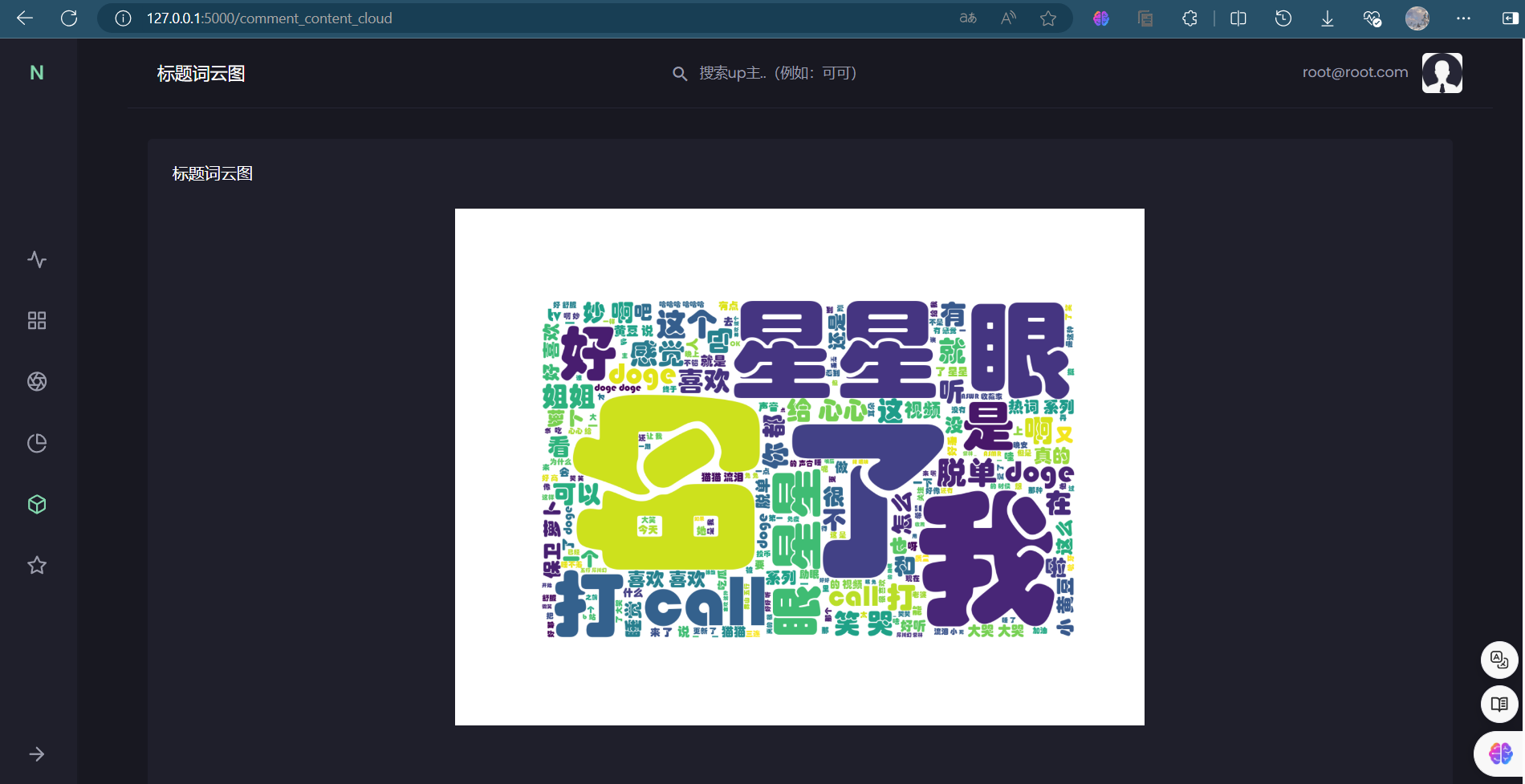
首先，通过在线模板修改，确定了文件的主题结构，并进行了相关内容的调整。随后，需要获取或生成需要展示的文本数据，然后在HTML页面中引入词云图生成的JavaScript库，将生成的词云图嵌入到页面中的适当位置，一个div容器。

通过添加交互功能，鼠标悬停在词语上显示详细信息或者点击词语进行相关搜索等，增强用户体验。

接着使用css文件的修饰功能，将该生成的词云图放置所需的路径下，进行页面展示，并且，设计了其他交互以确保用户的交互性得到保证。

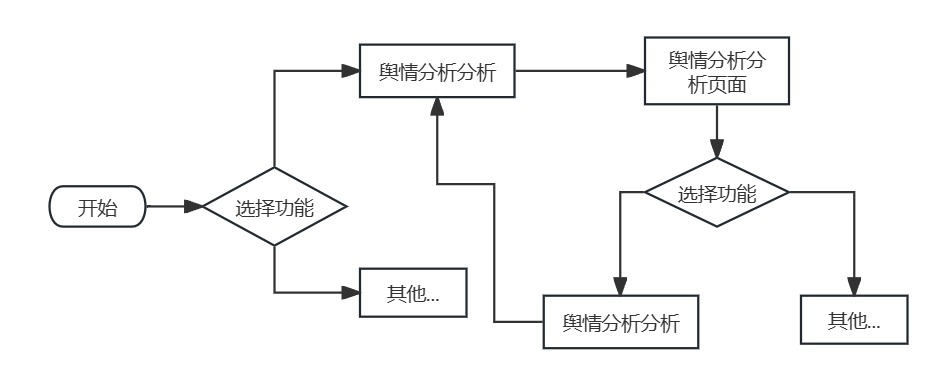
总之，这种设计不仅确保了用户信息的安全提交和良好的视觉体验，还通过JavaScript的支持，为页面增添了更多的交互性和动态效果，从而提升了整体的用户体验。

前端展示：

11.舆情分析模块

设计流程：



后端设计：

定义了一个路由处理函数，当用户访问'/ sentimentComment '路径时，该函数被调用。

会话和数据获取:

email = session['email'] 和 userRole = session['userRole']: 从会话中获取用户的电子邮件和角色信息。前提是在用户登录并通过身份验证后，这些信息被设置在会话中。

upInfoAllList = getPublicData.getAllUpInfo(): 调用getPublicData模块中的getAllUpInfo()函数，获取所有上行信息的列表。

时间处理:

timeList = getSentimentData.getTimeList(): 调用getSentimentData模块中的getTimeList()函数，获取时间列表。

defaultTime = request.args.get('time') if request.args.get('time') else timeList[0]: 从请求参数中获取时间参数'time'，如果没有则使用timeList中的第一个时间作为默认时间。

情感分析和数据处理:

x, y = getSentimentData.getSentimentCommentData(defaultTime): 根据默认时间获取情感评论数据，返回x轴和y轴数据。

dataList, content = themeModel.main(defaultTime): 根据默认时间调用themeModel模块的main()函数，返回数据列表和内容。

情感分析结果:

sentimentRes = '积极' if SnowNLP(content).sentiments >= 0.5 else '消极': 使用SnowNLP库对内容进行情感分析，如果情感得分大于等于0.5，则为积极，否则为消极。

数据处理为模板渲染准备:

resData: 将x轴和y轴数据组装成一个列表，每个元素包含名称和值。

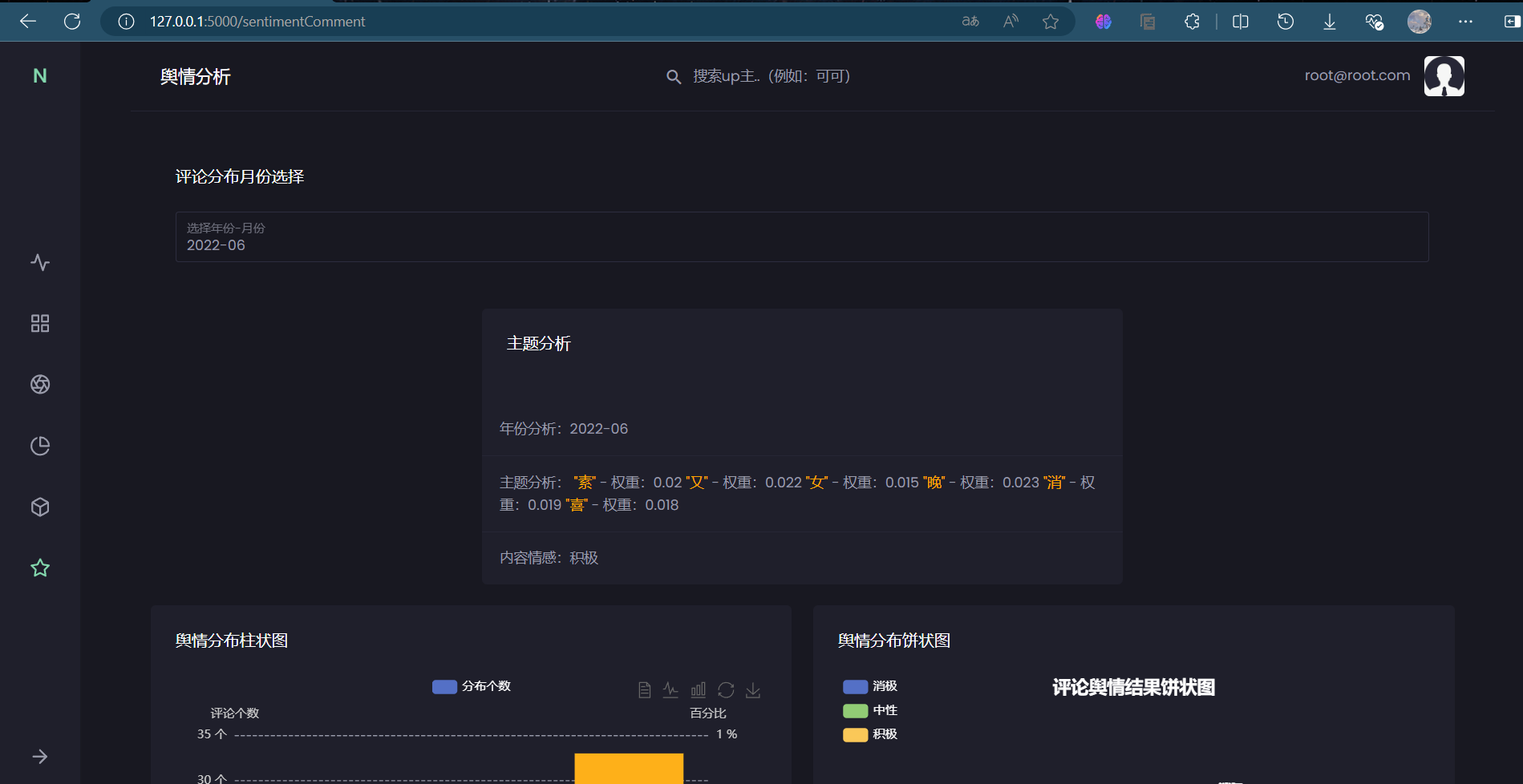
模板渲染:

render\_template('sentimentComment.html', ...): 最后，将所有数据传递给名为sentimentComment.html的模板，以便在模板中动态生成页面内容。

前端设计：

定义了一个路由处理函数，当用户访问'/sentimentComment’路径时, Flask 将调用这个函数来处理请求。从会话（session）中获取用户用户角色。通过函数调用调用放在utils中的辅助函数和工具函数获取首页需要展示的评论数据，如upInfoAllList = getPublicData.getAllUpInfo(): 调用getPublicData模块中的getAllUpInfo()函数，获取所有上行信息的列表。最后，使用 render\_template 函数将获取的数据传递给名为sentimentComment.html的模板文件。模板文件将使用这些数据来动态生成 HTML 页面，向用户展示视频分析页面的各种信息。

前端展示：

## 4.5 版本控制

### 4.5.1版本管理:

本项目使用Git进行版本控制：

1、设置版本控制系统：

安装并配置选定的版本控制系统Git。

在项目根目录下初始化版本控制，使用Git的命令 git init。

2、建立Github仓库：

上传初始版本项目，仓库地址：<https://github.com/yuluofengsu/Visual-analysis-of-B-station-comment-data-based-on-Flask.git>

3、加入仓库：

连接到GitHub上的远程仓库，使用 git remote 添加远程仓库

4、克隆项目到本地，分工开始编程：

各组员将本地的提交推送到远程仓库，使用 git push 命令。

各组员拉取远程仓库的更新到本地，使用 git pull 命令。

5、提交代码：

各组员分别将其负责模块代码上传，将其创建或修改的文件添加到版本控制中，使用 git add 命令。

### 4.5.2代码审查:

代码审查是项目开发过程中重要的一部分，通过检查、讨论和评估代码来确保代码质量和提高团队成员的编码技能的过程。以下是在本次项目开发中代码审查的过程：

准备：编码部分的组员完成编码并提交代码审查请求。

审查：其他小组成员仔细检查代码，查找错误、潜在的问题或改进机会。

讨论：编码部分的组员和审查部分的组员讨论发现的问题，并提供建议或修改建议。

记录：记录审查结果和建议的修改，确保问题得到解决或改进。

批准：通过审查后，审查部分的组员批同意将代码合并到主分支或下一阶段的开发中。

在本次项目开发中，通过代码审查，我们提高了代码的质量，通过发现和修复问题，减少了缺陷和错误的概率，促进了团队成员之间的知识传递和技术学习，增强团队合作精神和成就感。

## 4.6 功能测试

### 4.6.1测试计划:

1、用户登录/注册模块测试：

用户登录模块：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模块  编号 | 序号 | 测试用例描述 | 操作过程及数据 | 预期结果 | 验证结果 |
| 1.1 | 功能描述：登录功能 | | | | |
| 1 | 密码输错 | 邮箱：123456@qq.com  密码：admin | 登录失败，请检查密码 | √ |
| 2 | 账号输错 | 邮箱：12356@qq.com  密码：123456 | 登录失败，请检查用户名或密码 | √ |
| 3 | 正常登录 | 邮箱：123456@qq.com  密码：123456 | 进入系统 | √ |

用户注册模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 邮箱格式输错 | 邮箱：123456qq.com  密码：admin 确认密码：admin | 注册失败，请检查密码邮箱格式 | √ |
| 2 | 账号密码确认输错 | 邮箱：12356@qq.com  密码：123456 确认密码：12345 | 登录失败，请检查密码是否和确认密码一致 | √ |
| 3 | 正常注册 | 邮箱：123456@gmail.com  密码：123456 确认密码：123456 | 进入登录界面进行登录 | √ |

2、后台数据管理：

用户信息管理

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 账户修改 | 修改前：  邮箱：123456qq.com  密码：admin 角色：用户 修改后： 邮箱：123456qq.com  密码：123456 角色：用户 | 修改成功，该用户信息已被修改 | √ |
| 2 | 账户删除 | 修改前： 邮箱：12356@gmail.com  密码：123456 角色：用户 删除后： 均为空 | 删除成功，该用户在数据库中的信息也被删除，前端该用户无法进行登录 | √ |
| 3 | 数据库信息修改 | 对up主信息进行修改： 修改前：  Mid：238615690  Upname：助眠森林  Fensi: 8707  Et.  修改后： Mid：238615690  Upname：森森  Fensi: 8707  Et. | 信息修改成功，该修改会对数据库信息进行修改并对前端用户展现修改后的信息 | √ |
| 4 | 数据库信息删除 | 对评论信息进行删除： 修改前：  videoid：985121523  Uname：一直在努力的菜鸡沧海  content: 是无人声吗[星星眼]好赞 就喜欢无人声[打call][打call][打call]  Et.  修改后： 均为空 | 删除信息成功，该修改会对数据库进行删除操作，并将修改后的信息通过网页向前端进行展示 | √ |

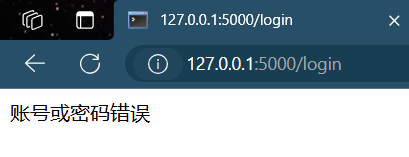
### 4.6.2测试执行:

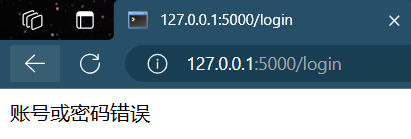
* 1. 用户登录模块测试：

登录前：



* 1. 密码输错：



* 1. 账号输错：



(3) 正常登录：

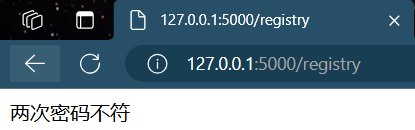
 

* 1. 注册模块测试：

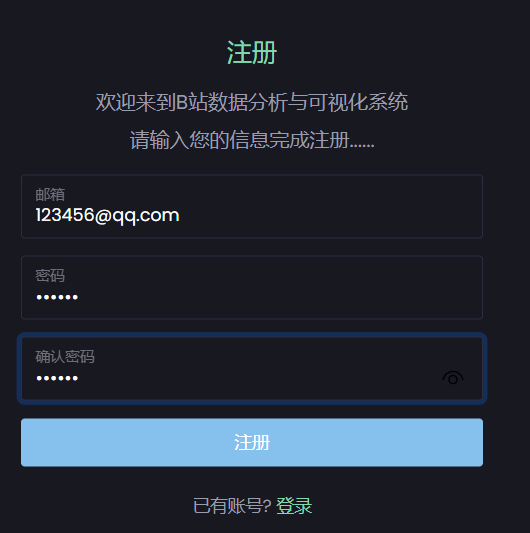
（1）邮箱格式输错、

（2）账户密码确认输错、

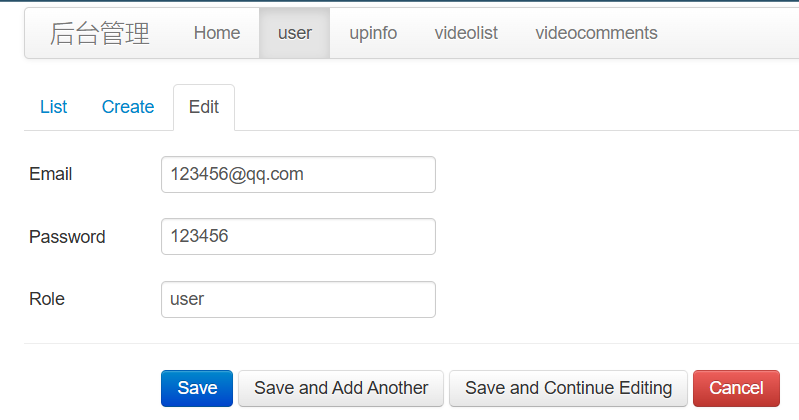
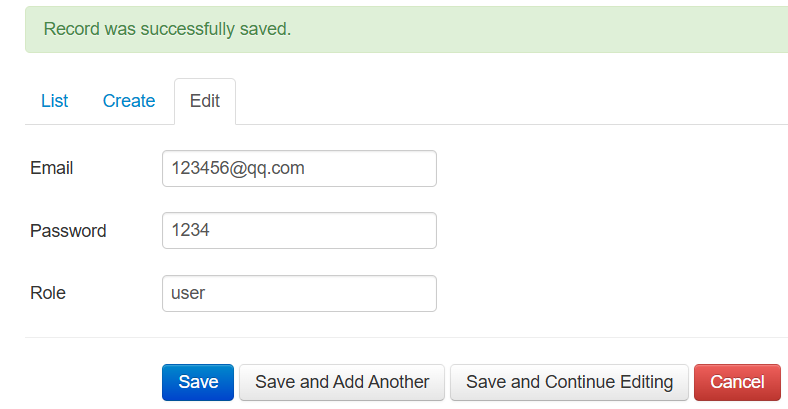
 

（3）正常注册、

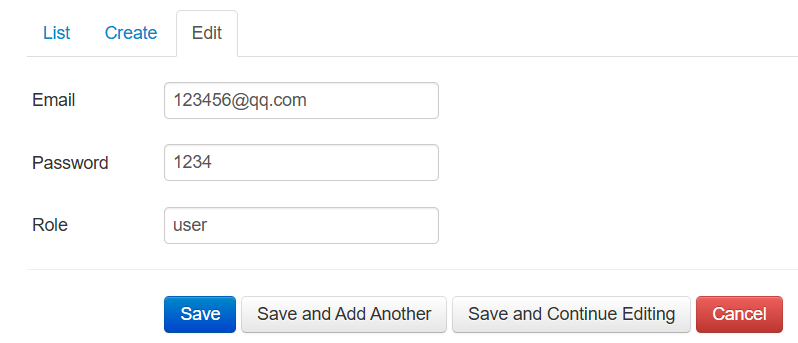
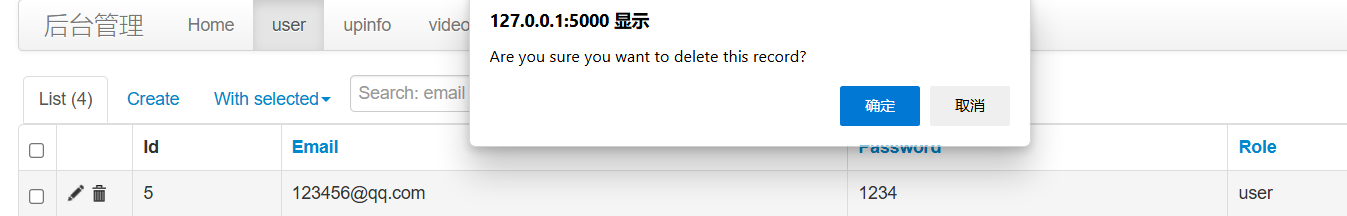
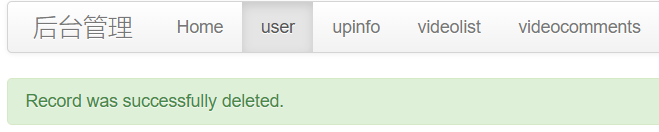
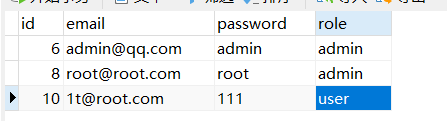
 

* 1. 后台数据管理：

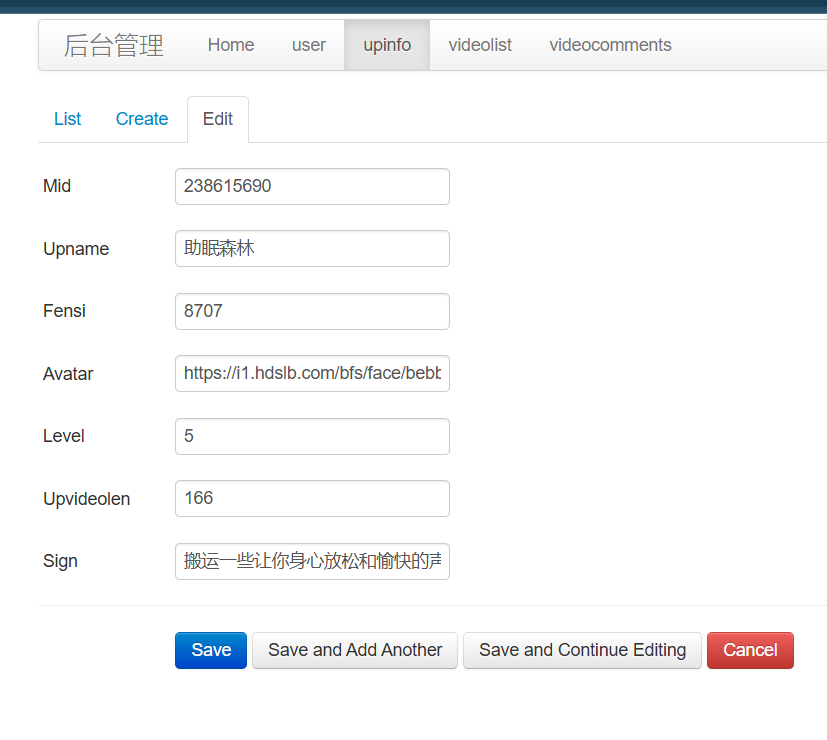
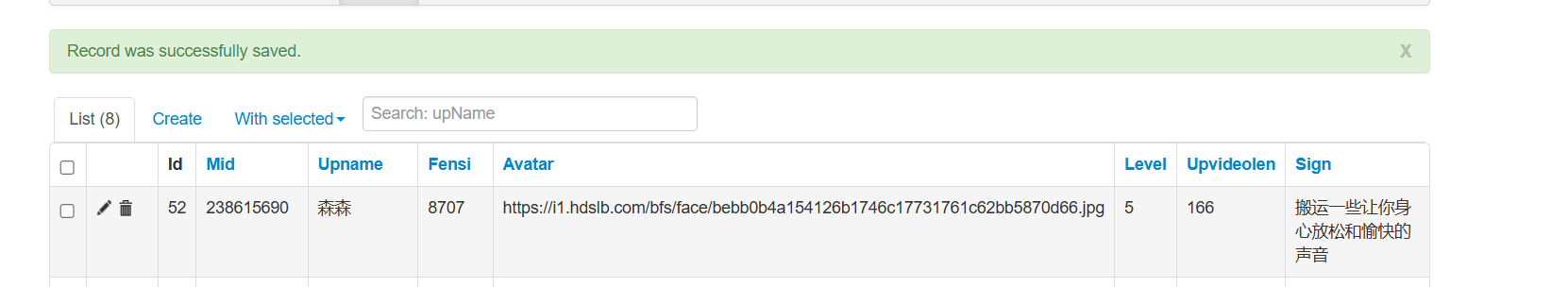
（1）账户修改：

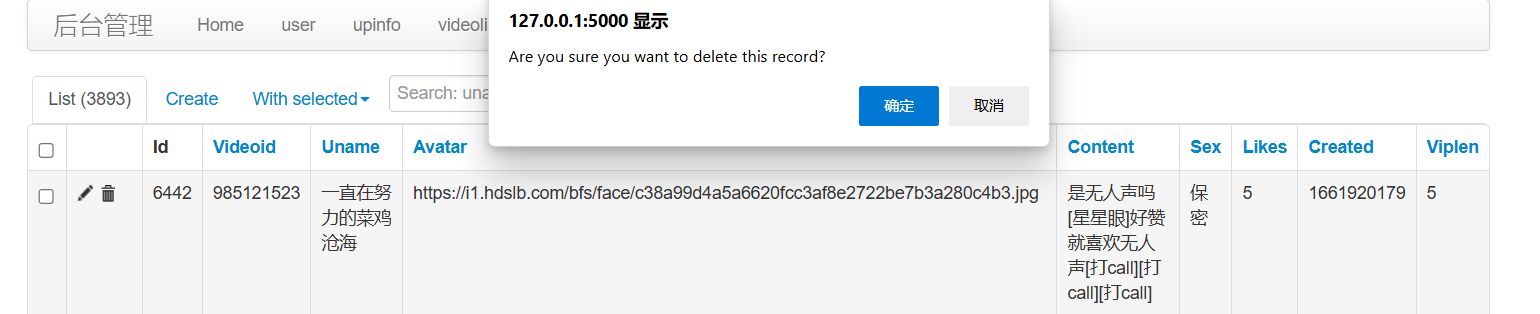
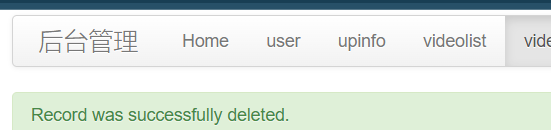
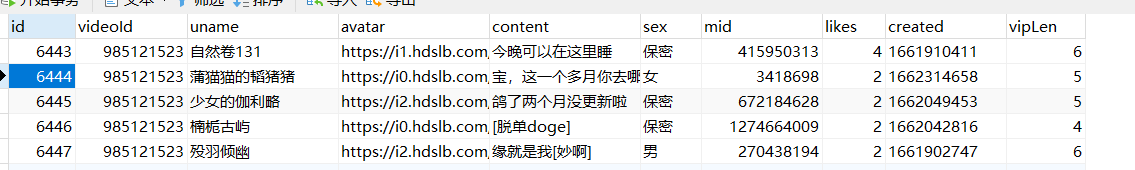
（2）账户删除：

（3）数据库信息修改：

（4）数据库信息删除：

问题：数据库连接上会出现问题，比如出现无法访问数据库的现象，此时需要根据报错代码进行修复，例如：数据库密码错误，数据库远程和本地密码不一致的错误。解决以上问题需要更改数据库密码，对数据库重新建立连接。

对数据库操作也会有问题，问题建立在数据库的连接上，主要问题是数据操作语句，对数据库进行一定的操作，出现问题之后，对数据操作语句进行修正即可完成错误修改。

## 4.7 问题解决

### 4.7.1问题记录:

1.项目结构混乱，不同模块之间的界限不明显

2.requests爬虫请求报错：UnicodeEncodeError: 'latin-1' codec can't encode character '\u2026' in position 30

3.requests 爬虫警告：InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate verif

4.pip版本不符，无法安装扩展包

### 4.7.2解决方案:

1.针对问题一，我们在原项目的基础上使用ORM技术重构项目，将数据库操作部分与主程序部分分离，使项目的结构更加清晰

2.当request请求中，带有中文，可能引发该报错，解决方法是，找到引起报错的地方（本项目中是在headers2中）把引起报错的字符串进行转码即可：headers2'.encode("utf-8").decode("latin1")

3.出现该错误的原因是由于requests 库是基于urllib 编写的，对urllib进行了封装，现在urllib已经出到了3版本，而requests最新版本也是基于最新的urllib3进行封装。在urllib2时代对https的处理非常简单，只需要在请求的时候加上verify=False 即可，但在urllib3时代，官方强制验证https的安全证书，如果没有通过是不能通过请求的，虽然添加忽略验证的参数，但是依然会给出醒目的Warning，不过实际上并不影响代码的正常运行。解决该问题只需要先找到原来请求网页的连接，在前面一行添加如下代码：

requests.packages.urllib3.disable\_warnings()

4.该错误是由于pip系统安装版本与项目版本不一致导致，可以先通过控制台指令pip list查看当前pip版本，之后将pip更新至最新版本即可，或者安装低版本的扩展包替代。

## 4.8 系统集成

在项目实现过程中，我们小组首先对数据分析项目展开了调查，对用户的需求进行了深入的分析，确保我们的系统设计能够满足目标用户的核心需求。在这个过程中我们定义了清晰的功能需求，为后续的设计和开发奠定了基础。

在设计阶段，我们采用了模块化的设计方法，将系统划分为前端用户界面、后端管理服务、数据库等模块。每个模块都有明确的职责和接口定义，便于小组成员分工合作。

在开发准备阶段，我们小组搭建了统一的开发环境，使用python3.7作为解释器，选择Pycharm作为开发工具，选择Mysql作为数据库，同时使用Git作为版本控制系统。在编码实现过程中，我们遵循了严格的编码规范，并定期进行代码审查，以保证代码质量和项目的顺利进展。

在功能测试阶段，我们小组制定了详细的测试计划，并通过自动化测试和手动测试相结合的方式，确保了每个功能模块的稳定性和可靠性。在问题解决环节，我们记录了遇到的问题，包括项目结构上的问题，底层代码设计上的问题等，并且迅速采取行动，通过团队协作找到了有效的解决方案。

在系统集成后，我们小组进行了全面的集成测试，包括用户登录、注册测试，可视化信息展示测试，后台信息管理测试等，确保了本系统作为一个整体的协调性和性能。在用户验收测试阶段，我们邀请了其他成员参与测试，并根据他们的反馈进行了一些调整，使得项目更加完善。

# 五、项目总结

# 附录