程序文档介绍

（1）系统设计文档

该系统基于 Flask 框架开发，分为前台和后台两个部分，涵盖了用户交互和管理操作。具体设计分为以下几个模块：

a.前台模块：

用户注册与登录：支持新用户注册与现有用户登录。

电影播放：用户可以观看电影，并进行评论、收藏等操作。

会员中心：用户可以查看个人资料、收藏的电影以及历史评论。

b.后台模块：

管理员登录：支持管理员登录与权限管理。

标签管理：管理员可以对电影标签进行增删改操作。

电影管理：支持电影的上传、修改和删除。

评论管理：管理员可以查看、删除用户的评论。

网站建议：用户提出网站建议，管理员可以处理这些建议。

数据库设计：

数据库采用 MySQL，包含多张表，如用户表、电影表、评论表、收藏表等。数据模型设计基于 ORM 工具（如 SQLAlchemy），通过模型与数据库的映射实现数据的增删改查。

系统架构：

```
Flask_movie_project/      <-- 项目根目录
├── app                   <-- 应用程序目录
│   ├── admin             <-- 后台功能模块
│   ├── home              <-- 前台功能模块
│   ├── models.py         <-- 数据模型
│   └── templates         <-- 前端页面模板
├── logs                  <-- 日志文件夹
├── manage.py             <-- 项目启动脚本
├── movie.sql             <-- 数据库建表 SQL 文件
└── req.txt               <-- 项目依赖
```

（2）说明文档

a.项目概述：

该项目为一个基于 Flask 框架开发的电影网站，用户可以在前台浏览和评论电影，同

时后台管理员可以对网站内容进行管理。该项目采用了经典的 MVC（Model-View-Controller）架构。

b.功能说明：

前台功能：

用户注册与登录：允许用户通过邮箱注册新账户，并登录管理个人信息。

电影播放：用户可以观看网站提供的电影资源，进行评论与收藏操作。

会员中心：用户登录后，可以查看个人信息，管理评论和收藏的电影。

后台功能：

标签管理：管理员可以管理电影的分类标签，方便用户筛选。

电影管理：管理员能够上传、修改电影信息，并管理电影的上映预告。

评论管理：管理员可管理用户的评论，删除不符合规定的评论内容。

用户管理：对注册用户的信息进行管理。

c.项目依赖**：

项目中依赖的包包括 Flask、SQLAlchemy 等，详细依赖列表保存在`req.txt`文件中，用户可通过安装该文件中的依赖来运行项目。

（3）使用文档

a.安装项目依赖：

首先，需要克隆项目到本地并安装项目所需的 Python 依赖包：

bash

```
git clone https://github.com/qinbin52qiul/Flask_movie_project.git
cd Flask_movie_project
pip install -r req.txt
```

b.导入数据库：

将项目附带的数据库导入到本地 MySQL 中：

bash

```
mysql -u 用户名 -p 数据库名 < movie.sql
```

c.启动项目：

运行以下命令启动 Flask 项目，默认在本地`127.0.0.1`和端口`5000`启动：

bash

```
python manage.py runserver -h 127.0.0.1 -p 5000
```

浏览器访问 http://101.42.16.53:5000/即可进入网站。

d.登录后台：

管理员登录后台进行管理，后台功能包括电影管理、评论管理、用户管理等。网址为 http://101.42.16.53:5000/admin/

（4）测试文档

a.功能测试：

用户注册与登录：测试新用户是否可以成功注册及登录。

电影播放：验证用户是否能够正常观看电影、发表评论与收藏电影。

后台管理功能：测试管理员是否能够正确登录并管理电影、标签、用户评论等内容。

b.性能测试：

并发用户测试：模拟多个用户同时登录、浏览电影，验证系统响应速度和稳定性。

数据库性能测试：测试系统在处理大量用户请求时，数据库的读写性能。

c.错误处理：

异常数据测：测试系统是否能正确处理用户输入的非法数据，如特殊字符、空数据等。

边界情况测试：验证系统在极端情况下（如用户登录失败次数过多、评论内容过长等）的稳定性。

(5）总结

该项目为一个完整的视频网站解决方案，涵盖了用户端的视频播放与互动功能，以及管理员的后台管理功能。通过 Flask 框架构建，结构清晰，功能完备，适合作为视频网站的基础开发框架。在实际应用中，可以通过扩展功能、优化性能来实现更复杂的视频平台需求。