

项目总结报告

|  |  |  |  |
| --- | --- | --- | --- |
| Prepared by  拟制 | 涂诗文，肖阳，余旭明  曹祥琛 | Date  日期 | 2024/12/28 |
| Reviewed by  评审人 | 涂诗文 | Date  日期 | 2024/12/29 |
| Approved by  批准 | 涂诗文 | Date  日期 | 2024/12/29 |

# 项目的基本情况

表 1：项目基本情况表

|  |  |  |  |
| --- | --- | --- | --- |
| **项目名称：** | 电力知识库与在线学习平台 | | |
| **项目类别：** | PythonWeb | | |
| **开发环境：** | **python版本** | v3.8 | |
| **Node版本** | v18.20.5 | |
| **测试工具** | Curl，openAPI | |
| **采用技术：** | **前端** | React，AntD | |
| **后端** | Django4.2.7 | |
| **数据库** | MySQL8.0 | |
| **运行平台：** | Ubuntu 24.4 | 项目起止时间： | 2024/10/25-2024/12/30 |
| **项目经理：** | 涂诗文 | | |
| **项目成员：** | 涂诗文 肖阳 余旭明 曹祥琛 | | |
| **项目描述：** | 有助于提升电力企业的培训效率和考核质量，加速人才培养和成长；同时也有助于推动电力行业的数字化转型和高质量发展。 | | |

# 项目的完成情况

当前项目完成度达到70%,本项目根据设计一共应有20个界面,只完成了15个。完成了前台在线课程功能，个人中心，课程评论区评论功能，后台实现了用户的权限分层，用户管理，课程管理，章节管理，评论管理，反馈管理。

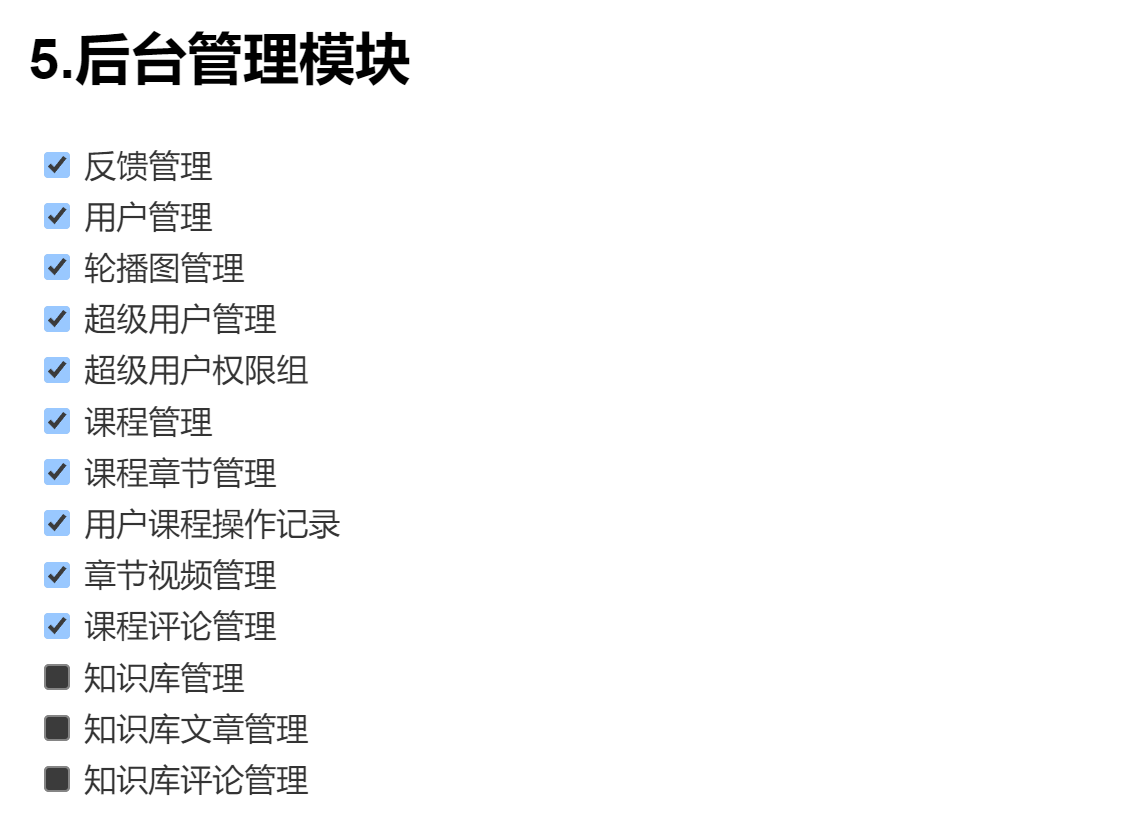
具体完成情况如下：











# 任务及工作量总结

表 2：项目个人贡献统计表

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 代码实现负责模块 | 文档编写负责模块 | 贡献比 |
| 涂诗文 | 前台前端，推荐课程和在线课程实现，后台课程，管理实现、后台用户，评论，反馈管理实现 | 项目立项、需求分析文档编写，项目总结文档编写，文档修订 | 32% |
| 肖阳 | 前台前端首页实现、网站内容设置 | 原型设计 | 25% |
| 余旭明 | 前台、后台系统测试、前台课程评论区实现 | ppt制作、测试用例文档撰写 | 22% |
| 曹祥琛 | 前台后端登录注册实现，数据库设计 | 系统设计文档编写，文档修订 | 21% |

表 3: 项目代码量统计表

|  |  |  |
| --- | --- | --- |
| 代码模块 | 代码行数/注释行数 | 统计指令 |
| Java文件 | **815** | find . -name "\*.java"|xargs cat|grep -v ^$|wc -l |
| JSx文件 | **339** | find . -name "\*.jsx"|xargs cat|grep -v ^$|wc -l |
| 合计 | **1154行** | |

# 四、项目总结

整个项目总共历时两个月，所有中间有一段时间暂时搁置，但是总体开发时间也较长，整个项目经历了项目立项，到需求分析，原型设计，系统设计，到最后进行编码实现，整个项目小组的所有成员都花费了大量的时间和精力投入当中。除了要求提交的需求分析，系统设计文档之外，我们小组内部为了统一规范，为了指定任务计划还制定了详细的功能待办列表，API接口文档。项目文档格式规范等等。 最终较为圆满地完成了初期指定项目构想。功能丰富程度和规范程度相较我之前写过的项目也有较大提高。

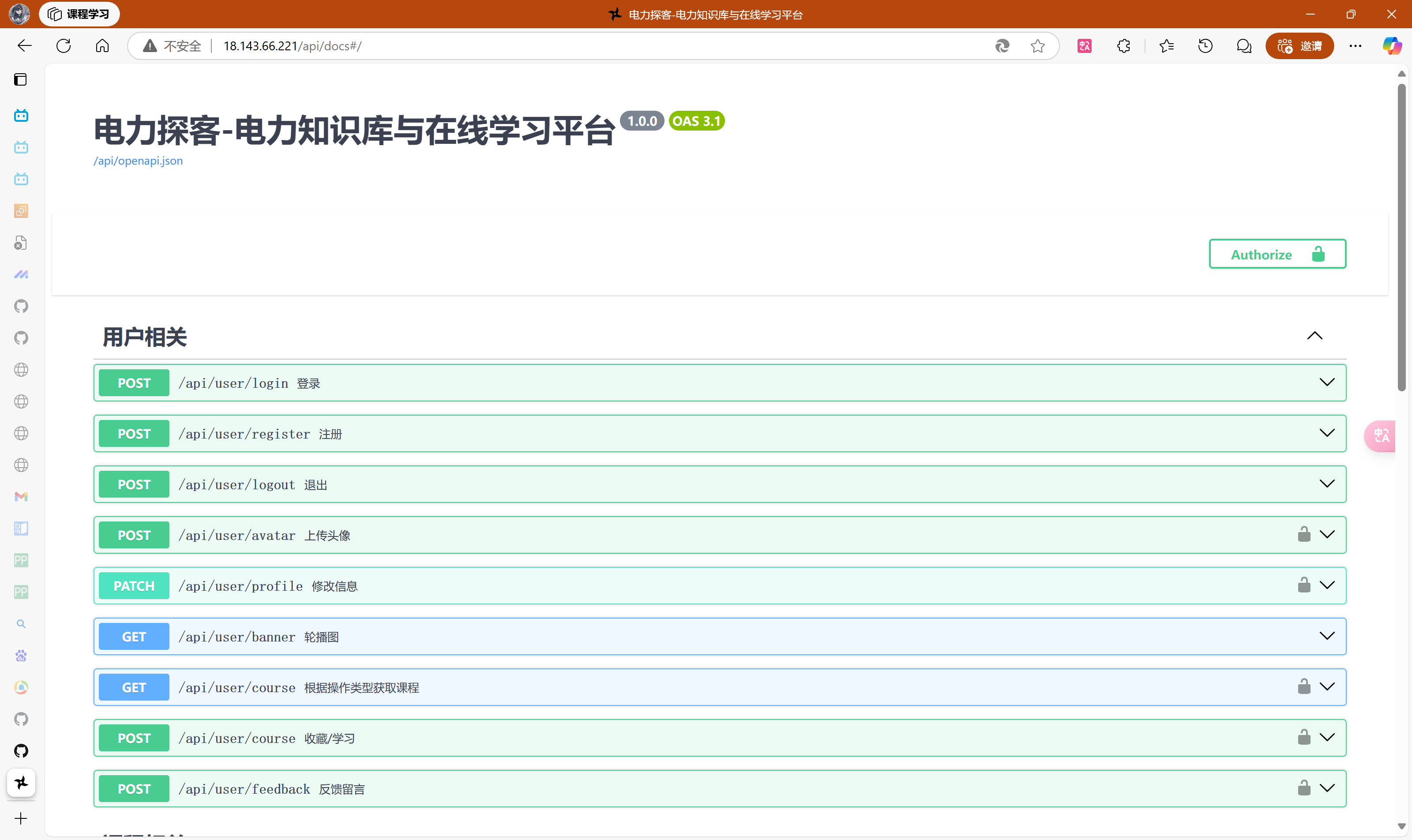


图1 API接口页面

由于我之前有过一次小型项目开发的经验，这次的再次使用三层架构（Controller、Service、Repository）进行开发，总体来说是较为熟练的。但是在开发工程当中还是出现了很多问题，最后通过此次项目我总结了几点经验：

1. **CSR的三层架构**

在进行 Django 项目开发时，我发现三层架构（Controller、Service、Repository）的重要性。初学者往往会在视图函数中处理过多的业务逻辑，这样会导致代码混乱，且不易维护。尤其是随着项目规模的扩大，代码量增加后，容易使得逻辑部分集中在视图层，最终使得代码难以重构和调试。因此，合理分层并将业务逻辑、数据访问和视图处理分离是非常关键的。尽量保持视图层的简洁，并将数据处理交由服务层（Service）和数据访问层（Repository）完成，可以提高代码的可读性和可维护性。

1. **Django ORM 和 React 组件使用**

在项目中，我们使用 Django 的 ORM 进行数据库操作，通常会用反向工程来自动生成模型，但在需求变化时，往往需要修改和增加表结构。一个常见问题是将多个数据库操作放在同一个视图函数中，这样会使得数据访问部分难以扩展和维护。为了避免这种情况，应该将不同的数据库操作和视图逻辑分开，确保每个操作的职责单一，并利用 Django ORM 提供的关系映射功能来优化数据库查询。

对于 React 前端开发，虽然框架本身提供了许多功能，但如果样式和功能过于自定义，可能会影响框架的表现。为了实现良好的界面效果，建议在使用框架前详细阅读官方文档，掌握框架的最佳实践，并尽量避免过多的自定义样式，以减少与框架默认样式的冲突。

1. **AOP的应用**

在开发中，常常需要处理身份验证和异常管理等重复性任务，这些逻辑在不同的视图中都会用到。为避免代码冗余，项目中采用了 AOP（面向切面编程）来统一处理这些公共功能。通过 AOP，我们可以将身份验证、权限检查和异常处理逻辑从核心业务代码中分离出来，使得代码更加简洁，并且提高了项目的可扩展性和可维护性。

1. **前端开发**

前端开发是整个项目中不可忽视的一部分，特别是在使用 React 框架时。尽管 React 提供了许多组件和功能，但如果随意混合自定义样式，可能会影响框架效果。因此，建议在开发前大量阅读官方文档，了解 React 的核心概念和推荐的最佳实践。只有清楚框架的使用方式，才能更好地发挥其优势，避免由于错误的使用方式导致效果不佳。

1. **团队合作**

作为项目的组长，我深知团队合作中的重要性。在项目进行中，我们团队共四人，涉及到任务分配、时间管理和贡献评估等方面。最常见的问题是个别组员的工作量过大，可能导致他们承担了大部分任务，而其他组员则未能有效参与。在我的管理中，我尽量保证每位组员都有机会参与到项目的各个环节，并且能从中学到东西，而不是单纯为了任务的完成而将某些人“边缘化”。确保每个人都有机会贡献自己的力量，这不仅能提升团队的整体协作效率，还能帮助每个成员成长。